

Title of the Thesis

Towards Alternative Characterizations of NP-Complete Sets

A Thesis Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

by

Manindra Agrawal

to the

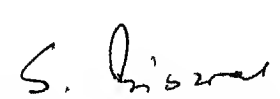
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR**

OCTOBER, 1991



CERTIFICATE

It is certified that the work contained in the thesis entitled "Towards Alternative Characterizations of NP-Complete Sets", by Manindra Agrawal, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.



(Signature of Supervisor)

S. Biswas

Professor

Department of Computer Science and Engg.

Indian Institute of Technology, Kanpur

Kanpur-208016, INDIA

October, 1991

CSE-1PRI-D-AGK-TOW

20 OCT 1953 / CSE

CENTRAL LIBRARY
U. S. AIR FORCE

Doc. No. A. 116870

Th

005.131

Ag 812

SYNOPSIS

Introduction

The present thesis aims at providing alternative characterizations of NP-complete sets. We approach this problem in two different ways — the first one is motivated by recursion theory while the second one has arisen out of combinatorial considerations of reductions amongst *natural* NP-complete sets. Using these approaches, we have found two sufficient properties for a set to be NP-complete. All known NP-complete sets *seem* to satisfy both of these properties, therefore, either may be considered as a possible alternative characterization of NP-complete sets. However, any proof showing either to be a *necessary* condition would immediately imply $P \neq NP$. In the following, we separately describe these two properties and major results concerning them.

(p, CM_{NP}) -Creative Sets

Creative and productive sets have been extensively studied in recursion theory. Productive sets, informally, are those non-r.e. sets that have a *constructive* proof of their non-r.e.-ness and creative sets are those r.e. sets whose complement set is productive. Creative sets turn out to be equivalent to many-one complete sets for the class of r.e. sets, thus giving a simple and elegant characterization of many-one complete sets. These two notions have been translated to bounded complexity classes as well.

Creativeness of a language over a language class is defined by specifying a function over a set of TM indices presenting the language class. While defining creativeness for bounded complexity classes, one encounters the following problem — depending

on the index set chosen to present a language class, the complexity of the creative set defined over this class may vary, e.g., while Ko and Moore¹ show that there are no creative sets over P in EXP , Wang² shows, using a different indexing of polynomial time languages, that there *are* creative sets over P in EXP . In fact, we show that by considering different index sets presenting a trivial class of languages, called $CONST$, one can obtain complete sets for different classes, ranging from $DLOG$ to $r.e.$! This suggests the following new scheme of obtaining creativeness for a class \mathcal{C} : fix a *suitable* index set presenting the class $CONST$ and then a language in \mathcal{C} is defined to be creative if it is creative over this index set. For the class EXP and beyond, the new definitions thus obtained turn out to be equivalent to old ones. However, for the class NP , the class we are interested in, this does not seem to be the case.

The existing definition of creativeness for NP is that of k -creativeness, given by Joseph and Young³. They have shown that such creative sets are NP -complete. However, it is not known if natural NP -complete sets are creative (except for the Bounded Tiling problem) and further, it is believed that they are not. Therefore, this definition of creativeness for NP does not serve our purpose of characterizing NP -complete sets.

We have explored the definition of creativeness for NP suggested by the scheme outlined above. We call such sets (p, CM_{NP}) -creative sets. This notion is a generalization of k -creativeness and it satisfies the following major properties.

1. If A is (p, CM_{NP}) -creative then A is NP -complete.
2. If A is k -creative then A is (p, CM_{NP}) -creative.

¹K. Ko and D. Moore, *Completeness, approximation and density*, SIAM J.Comput., 10(1981) 787-796.

²J. Wang, *P-creative sets vs. p-completely creative sets*, SCT(1989) 24-33

³D. Joseph and P. Young, *Some remarks on witness functions for nonpolynomial and noncomplete sets in NP*, TCS 39(1985) 225-237

3. If A is NP-complete and *lpr-paddable* then A is (p, CM_{NP}) -creative. (A set is *lpr-paddable* if it is paddable in a very direct and natural way, as is the case with natural NP-complete problems.)
4. If all paddable NP-complete sets are (p, CM_{NP}) -creative then all NP-complete sets are (p, CM_{NP}) -creative.

These results appear to suggest that all NP-complete sets are (p, CM_{NP}) -creative. However, proving that it is indeed the case will be difficult as this will immediately imply $P \neq NP$. (This follows from the fact that finite sets are not (p, CM_{NP}) -creative.)

In recursion theory, creativeness property has been used to prove a fundamental result : many-one complete degree forms a single isomorphism degree. For NP, the question of p-isomorphism of NP-complete sets is open. We tried to use the (p, CM_{NP}) -creativity to attack this question, but could not obtain any significant result. However, for a smaller class of reductions, called 1-L reductions, we have shown, using (p, CM_{NP}) -creativity, that all 1-L complete sets for NP are p-isomorphic. This result is, to the best of our knowledge, the first result of its kind for the class NP.

We have briefly considered creative sets for the classes EXP, NEXP and PSPACE as well. For EXP and NEXP, we show that our definitions of creative sets are equivalent to the earlier ones given by Wang. We also prove that creative sets for these classes characterize many-one complete sets. For the class PSPACE, we show that creative sets characterize logspace-complete sets and moreover, all such sets are complete under one-one and size-increasing reductions.

Universal Relations

When one examines the usual reductions amongst *natural* NP-complete sets, one finds that these reductions satisfy several nice properties; for example, they usually preserve

not only the membership, but also the number of *solutions*. (For set A in NP, suppose R is a polynomially verifiable binary relation such that $x \in A$ iff $(\exists y)[|y| \leq p(|x|) \wedge xRy]$ for some polynomial p . We call y 's in the above as *solutions* of the instance x .) Further, the instances in the range of such reductions are also highly structured : they usually consist of several copies of some instance joined in some regular ways. These considerations have led us to another property which implies NP-completeness.

In particular, we show that if corresponding to a relation for a set in NP, there exist two operators and a certain kind of instance defined for the set then the set will be NP-complete. We call the two operators as *join* and *equivalence*. The *join* operator takes two instances and produces, in polynomial time, an instance such that the solution set of the produced instance, after some modifications, is the *product* of the solution sets of the input instances. The *equivalence* operator takes an instance and two numbers and produces, in polynomial time, an instance such that the solution set of the produced instance, after some modifications, is equal to the set of all those solutions of the input instance in which the two bit positions given by the two numbers have the *same value*. For an example of these two operators, consider the standard relation for SAT in which the solution set of an instance is all satisfying assignments to it. Here, $join(x, y) \stackrel{\text{def}}{=} x \wedge y'$, where y' is obtained from y by renaming all its variables to make them different from those of x , and $equ(x, i, j) \stackrel{\text{def}}{=} x \wedge (v_i \vee \bar{v}_j) \wedge (\bar{v}_i \vee v_j)$, where v_i and v_j denote the i^{th} and j^{th} variables of x respectively.

The *join* and *equivalence* operators are closely related to paddability and d-self-reducibility respectively. We show that with some additional constraints, if a relation has a *join* operator then the witnessed set is paddable and if it has an *equivalence* operator then the witnessed set is d-self-reducible.

We call a relation *universal* if it has the above two operators satisfying some further properties (details given in the thesis) and an instance, called *building block*, having

a particular kind of solution set. We show that if a set is witnessed by a universal relation then it is NP-complete. The proof reduces any instance of 3SAT by taking a number of copies of the *building block*, joining them using the operator *join* iteratively to get a single instance and then restricting the solution set of this instance using the *equivalence* operator a number of times.

The major results concerning the universal relations are as follows.

1. If R is a universal relation for a set A then A is NP-complete via size-increasing reductions.
2. Every paddable NP-complete set has a universal relation.
3. Well known classes of k -creative sets in NP have universal relations.

This suggests a characterization, namely that all NP-complete sets have universal relations. However, this also implies $P \neq NP$ as it can be shown that finite sets can not have universal relations.

We show the standard relations for a number of NP-complete problems to be universal. In fact, as is the case with SAT, the construction of the two operators for them turns out to be fairly straightforward. We also exhibit relations for NP-complete sets that are non-universal. We generalize our definition to *generalized universal relations* and show that such non-universal relations are generalized universal. In fact, we did not find a relation for an NP-complete set which is not generalized universal. Therefore, if a set is witnessed by a relation which is not generalized universal, it may be taken as an evidence for non-completeness of the set. We prove that the standard relations for the Horn Clause and the Graph Isomorphism problems are not generalized universal.

To
Amma & Pitaji

ACKNOWLEDGEMENTS

I am indebted to my thesis supervisor Prof. S. Biswas for inspiring me to work on NP-complete sets. I am also thankful to him for teaching me, with partial success, how to write documents that others too can understand.

My ideas about the world would not be what they are but for Brajesh Pandey, Vijayan Rajan, Siddhartha Debgupta and Dhirendra Tripathi. And my life in IIT would not have been what it was but for the company of Subir Bhaumik, Rajesh Nagpal, Sanjay Bhargava, R.P. Singh, Anoop Chawla, Yudhvir Singh Parmar, Jishnu Biswas, Rohit Mittal, Prashant Adhikari and Navdeep Sood.

Manindra Agrawal

Contents

1	Introduction	1
2	Creative Sets Over Constant Time Languages	7
2.1	Introduction	7
2.2	Preliminaries	9
2.2.1	Notations	9
2.2.2	Basic definitions	10
2.3	A new definition of creativeness for NP	13
2.3.1	(p, CM_{NP}) -creative sets are NP-hard	13
2.3.2	k -Creative sets and (p, CM_{NP}) -creative sets	16
2.4	A new reducibility, preserving (p, CM_{NP}) -creativeness	17
2.5	The \leq_m^{pr} -reducibility	22
2.5.1	A new kind of padding function	22
2.5.2	Pr-simple sets	26
3	Further Results on Creative Sets	33
3.1	Introduction	33
3.2	Creative sets in NP	34
3.2.1	Polynomial isomorphism and (p, CM_{NP}) -creative sets	34

3.2.2	Productive sets and diagonalization	42
3.3	Creative sets in EXP, NEXP and r.e.	47
3.4	Creative sets in PSPACE	50
4	The Universal Relation	55
4.1	Introduction	55
4.2	Notations	56
4.3	Universal relations	58
4.4	Universal relations witness NP-complete sets	63
4.5	Some results useful in application	66
4.6	Examples of universal relation	68
4.6.1	Natural languages	68
4.6.2	k -Creative sets	73
4.7	Examples of non-universal relations	74
4.8	Structural properties of the two operators	80
4.8.1	The Join operator	80
4.8.2	The Equivalence operator	82
4.9	The universal and generalized universal relations	85
5	Concluding Remarks	88
	Index	93

List of Figures

4.1	The modified part of the graph output by the negation operator for R_{HAM}	70
4.2	The instance $block_{R_{HAM}}$	71
4.3	Graph for the instance $block_{R_{CLIQUE}}$	72
4.4	Graph for the instance $block_{R'_{SMC}}$	80

Chapter 1

Introduction

The present thesis aims at providing alternative characterizations of NP-complete sets. We approach this problem in two different ways — the first one is motivated by recursion theory while the second one has arisen out of combinatorial considerations of reductions amongst *natural* NP-complete sets. Using these approaches, we have found two sufficient properties for a set to be NP-complete. All known NP-complete sets *seem* to satisfy both of these properties, therefore, either may be considered as a possible alternative characterization of NP-complete sets. However, any proof showing either to be a *necessary* condition would immediately imply $P \neq NP$. In the following, we separately describe these two properties and major results concerning them.

(p, CM_{NP}) -Creative Sets

Creative and productive sets have been extensively studied in recursion theory [13]. Productive sets, informally, are those non-r.e. sets that have a *constructive* proof of their non-r.e.-ness and creative sets are those r.e. sets whose complements are productive. Creative sets turn out to be equivalent to many-one complete sets for the class of

r.e. sets, thus giving a simple and elegant characterization of many-one complete sets. These two notions have been translated to bounded complexity classes as well. Further, we shall see that for some of these classes too, polynomial time many-one completeness can be shown to be equivalent to creativeness.

Creativeness of a set over a language class is defined by specifying a function over a set of TM indices presenting the language class. This function is called the *productive function* for the set and it maps each index i in the TM presentation of the language class to the symmetric difference of the language W_i and the complement of the set. This ensures that the complement of the set, viz., the productive set, is not in the language class. While defining creativeness for bounded complexity classes, one encounters the following problem — depending on the index set chosen to present a language class, the complexity of the creative set defined over this class may vary, e.g., while Ko and Moore [11] show that there are no creative sets over P in EXP, Wang [14] shows, using a different indexing of polynomial time languages, that there are creative sets over P in EXP. In fact we discovered that the complexity of the creative set can be made to vary widely by choosing different indexings of the same language class. In chapter 2, we define a trivial class of languages called CONST, $\text{CONST} \stackrel{\text{def}}{=} \{L \mid L \text{ is accepted in constant time by some TM}\}$, and show that by considering different index sets presenting CONST, one can obtain complete sets for different classes, ranging from DLOG to r.e. ! An index set presenting the class CONST is $I = \{i \mid (\forall x)[T_{M_i}(x) \leq c_i]\}$, where c_i depends only on i . By varying c_i we obtain complete sets for several classes, e.g., the creative sets over I with $c_i = |i|$ are NP-hard. This leads us to a new definition of creativeness for NP; we examine in detail if this notion of creativeness for NP can give us an alternative characterization of NP-complete sets. We note here that the existing definition of creativeness for NP, namely k -creativeness [10], does not help in our aim as it is not known if natural NP-complete

sets are k -creative (except for the Bounded Tiling problem [7]).

In chapter 2, we explore this new definition of creativeness for NP, called (p, CM_{NP}) -creativity. The main results are :

1. If A is (p, CM_{NP}) -creative then A is NP-complete.
2. If A is k -creative then A is (p, CM_{NP}) -creative.
3. If A is NP-complete and *lpr-paddable* then A is (p, CM_{NP}) -creative. (*Lpr-paddability* is a new notion of paddability that we have defined (chapter 2). A set is *lpr-paddable* if it is paddable in a very direct and natural way, as is the case with natural NP-complete problems.)
4. If all paddable NP-complete sets are (p, CM_{NP}) -creative then all NP-complete sets are (p, CM_{NP}) -creative.

We have also defined in chapter 2 a new reducibility, called \leq_m^{pr} -reducibility, which is a restriction of \leq_m^p -reducibility. We show that all \leq_m^{pr} -complete sets for NP are *lpr-paddable*. Using the concept of simple sets [13], we also show that there is a kind of simple sets in NP-complete degree that are not *lpr-paddable*. However, these sets are also shown to be (p, CM_{NP}) -creative.

These results appear to suggest that all NP-complete sets are (p, CM_{NP}) -creative. However, proving that it is indeed the case will be difficult as this will immediately imply $P \neq NP$. (This follows from the fact that finite sets are not (p, CM_{NP}) -creative.)

In recursion theory, creativeness property has been used to prove a fundamental result : the many-one complete degree forms a single isomorphism degree. It is, therefore, natural to ask whether the notion of (p, CM_{NP}) -creativity can play a similar role in NP. Though we have no results bearing on the p -isomorphism question in its entirety, we do show in chapter 3 that for a smaller class of reductions, called 1-L

reductions, all 1-L complete sets for NP are p-isomorphic. This result is, to the best of our knowledge, the first result of its kind for the class NP. The result can be easily generalized to most of the standard complexity classes as well.

In chapter 3, we have also considered briefly creative sets for the classes EXP, NEXP and PSPACE. We first obtain definitions for them by choosing suitable index set presenting the class CONST. For EXP and NEXP, we show that our definitions of creative sets are equivalent to the earlier ones given by Wang [14]. We also prove that creative sets for these classes characterize many-one complete sets. For the class PSPACE, we show that creative sets characterize logspace-complete sets and moreover, all such sets are complete under one-one and size-increasing reductions.

Universal Relations

When one examines the usual reductions amongst *natural* NP-complete sets, one finds that these reductions satisfy several nice properties; for example, they usually preserve not only the membership, but also the number of *solutions*. (For set A in NP, suppose R is a polynomial time verifiable binary relation such that $x \in A$ iff $(\exists y)[|y| \leq p(|x|) \wedge xRy]$ for some polynomial p . We call y 's in the above as *solutions* of the instance x .) Further, the instances in the range of such reductions are also highly structured : they usually consist of several copies of some instance joined in some regular ways. These considerations have led us to another property which implies NP-completeness.

In particular, we show that if corresponding to a relation for a set in NP, there exist two operators and a certain kind of instance defined for the set then the set will be NP-complete. We call the two operators as *join* and *equivalence*. The *join* operator takes two instances and produces, in polynomial time, an instance such that the solution set of the produced instance, after some modifications, is the *product* of

the solution sets of the input instances. The *equivalence* operator takes an instance and two numbers and produces, in polynomial time, an instance such that the solution set of the produced instance, after some modifications, is equal to the set of all those solutions of the input instance in which the two bit positions given by the two numbers have the *same value*. For an example of these two operators, consider the standard relation for SAT in which the solution set of an instance is all satisfying assignments to it. Here, $join(x, y) \stackrel{\text{def}}{=} x \wedge y'$, where y' is obtained from y by renaming all its variables to make them different from those of x , and $equivalence(x, i, j) \stackrel{\text{def}}{=} x \wedge (v_i \vee \bar{v}_j) \wedge (\bar{v}_i \vee v_j)$, where v_i and v_j denote the i^{th} and j^{th} variables of x respectively.

The *join* and *equivalence* operators are closely related to paddability and d-self-reducibility respectively. We show that with some additional constraints, if a relation has a *join* operator then the witnessed set is paddable and if it has an *equivalence* operator then the witnessed set is d-self-reducible.

We call a relation *universal* if it has the above two operators satisfying some further properties (details given in chapter 4) and an instance, called *building block*, having a particular kind of solution set. We show that if a set is witnessed by a universal relation then it is NP-complete. The proof reduces any instance of 3SAT by taking a number of copies of the *building block*, joining them using the operator *join* iteratively to get a single instance and then restricting the solution set of this instance using the *equivalence* operator a number of times.

The major results concerning the universal relations are as follows.

1. If R is a universal relation for a set A then A is NP-complete via size-increasing reductions.
2. Every paddable NP-complete set has a universal relation.
3. Well known classes of k -creative sets in NP have universal relations.

This suggests a characterization, namely that all NP-complete sets have universal relations. However, this also implies $P \neq NP$ as it can be shown that finite sets can not have universal relations.

We show the standard relations for a number of NP-complete problems to be universal. In fact, as is the case with SAT, the construction of the two operators for them turns out to be fairly straightforward. We also exhibit relations for NP-complete sets that are non-universal. We generalize our definition to *generalized universal relations* and show that such non-universal relations are generalized universal. In fact, we did not find a relation for an NP-complete set which is not generalized universal. Therefore, if a set is witnessed by a relation which is not generalized universal, it may be taken as an evidence for non-completeness of the set. We prove that the standard relations for the Horn Clause and the Graph Isomorphism problems are not generalized universal.

Chapter 2

Creative Sets Over Constant Time Languages

2.1 Introduction

We explore in this chapter a new notion of creativeness in NP in our attempt towards an alternative characterization of NP-complete sets.

Creative sets over a language class are defined by specifying a function over a TM presentation of the language class. As a result the complexity of creative sets over a language class \mathcal{C} , of bounded complexity, does not depend solely on \mathcal{C} but also on the index set chosen to present the class. In fact, we define a very trivial language class called CONST and obtain, by varying the index set presenting the class, creative sets over CONST which range from DLOG-complete to r.e.-complete. Considering a suitable presentation of CONST, we also obtain a new definition of creativeness for the class NP, called (p, CM_{NP}) -creativeness, and then see whether this gives a characterization of NP-complete sets.

We first prove that all (p, CM_{NP}) -creative sets in NP are NP-complete, thus show-

ing that the property is sufficient for NP-completeness. As there are sets in P that are trivially not (p, CM_{NP}) -creative, a proof that the property is necessary for NP-completeness implies $P \neq NP$. Even after assuming $P \neq NP$, we could not prove the converse. However, we do the next best thing : we show that all existing NP-complete sets seem to be (p, CM_{NP}) -creative. Natural complete sets form the most well known subclass of NP-complete sets. To show them (p, CM_{NP}) -creative, we define a restriction of \leq_m^p -reducibility denoted as \leq_m^{pr} -reducibility. The property of transducers computing such reductions is that they must output bits ‘regularly’, the precise sense of which is explained later. We show that all \leq_m^{pr} -complete sets for NP are (p, CM_{NP}) -creative; and moreover, these are precisely those complete sets that are paddable in a very easy and direct way as is the case with natural complete sets. Thus, it provides strong evidence that all natural complete sets are (p, CM_{NP}) -creative. However, not all NP-complete sets are \leq_m^{pr} -complete. This is shown by following the analogy of r.e. simple sets; we exhibit a type of simple sets in NP-complete degree that are not \leq_m^{pr} -complete. However, these simple sets also turn out to be (p, CM_{NP}) -creative. The other well known class of NP-complete sets is that of k -creative sets (see [10] and [14]). We show that these sets are trivially (p, CM_{NP}) -creative.

The chapter is organized as follows. Section 2.2 gives the notations and definitions we use. In section 2.3, we define (p, CM_{NP}) -creativity and compare it with k -creativity. We also prove that every (p, CM_{NP}) -creative in NP is NP-complete. Section 2.4 introduces \leq_m^{pr} -reducibility and relates it to (p, CM_{NP}) -creativity. In section 2.5, we define lpr-paddable sets and show that all \leq_m^{pr} -complete sets for NP are lpr-paddable. We also observe that all natural sets appear to be lpr-paddable. We then go on to define pr-simple sets and using them show that \leq_m^{pr} - and \leq_m^p -degrees differ everywhere.

2.2 Preliminaries

2.2.1 Notations

Languages are defined over the alphabet $\Sigma = \{0,1\}$. Sometimes we refer to strings over $\{0,1,\#\}$, which should be taken as being over $\{00,01,11\}$. The set $\Sigma_{op\ n}$ denotes the set of strings $\{s \mid s \in \Sigma^* \ \& \ |s| \ op \ n\}$ where $op \in \{=, >, <, \leq, \geq\}$. Similarly for any set A , the set $\{A\}_{op\ n}$ denotes the set $\{s \mid s \in A \ \& \ |s| \ op \ n\}$ where op is as before. Let $\langle \cdot, \cdot \rangle$ denote the standard pairing function $\langle n, m \rangle = 1/2 * (n + m) * (n + m + 1) + m$. Multiple work-tape Turing machines (TMs) with a read-only input tape and a one-way write-only output tape are our model for computation. Unless otherwise stated, our machines are non-deterministic. We consider a fixed encoding scheme of such machines into strings with the proviso that the *information bearing part of the code of a TM begin and end with a 1*. This property of the indexing scheme is formally stated as follows,

$$(\forall i)(\forall m)(\forall n) [\phi_i = \phi_{0^m i 0^n} \ \& \ (\forall x) [T_{M_{0^m i 0^n}}(x) = T_{M_i}(x)]]$$

using the notation given below. We shall refer to this property as the *paddability* property of the indexing scheme.

NOTE : Although we make use of the above property extensively, the proofs do not critically depend on this indexing scheme. It has been chosen only because it makes many proofs simpler. All the theorems in this paper will hold for any reasonable indexing scheme as the required property can be established by padding the index by non-reachable states also.

TMs are indexed by the strings that encode them. For any string i ,

M_i denotes the TM whose code is i .

$T_{M_i}(x)$ is the computation time (as defined in [8]) of M_i on input x .

$S_{M_i}(x)$ is the number of cells of work tape(s) used in the computation of M_i on x .

W_i is the subset of Σ^* accepted by M_i .

ϕ_i denotes the partial function from Σ^* to Σ^* computed by M_i .

$\phi_i^t(x)$ is the (possibly partial) output of TM M_i on input x in t steps.

For a set of TM indices I , let $\mathcal{L}(I) \stackrel{\text{def}}{=} \{W_i \mid i \in I\}$. $\mathcal{L}(I)$ is called the class of languages *presented* by I .

p_1, p_2, p_3, \dots is the sequence of polynomial functions with $p_i \stackrel{\text{def}}{=} \lambda n. (n^{|i|} + 2^{|i|})$. Note that this enumeration of polynomials is different from the standard one which defines $p_i \stackrel{\text{def}}{=} \lambda n. (n^i + 2^i)$.

In the proofs below, we use c_1, c_2, c_3, \dots to denote constants, unless otherwise specified.

Using the paddability property of the indexing scheme, we define a function which increases the size of a TM index without changing the TM : $Pad(n, i) \stackrel{\text{def}}{=} i0^{|n|}$; $Pad(n, i)$ yields an index which codes the same TM as index i and $Pad(n, i) > n$.

2.2.2 Basic definitions

Set $A (\leq_{m,1}^p, \leq_{m,si}^p, \leq_{m,i}^p) \leq_m^p B$ if A is reducible to B via a (one-one, size increasing, invertible) polynomial time function. For any set A , let $\mathbf{d}_m(A)$ denote its \leq_m^p -degree and $\mathbf{d}_{iso}(A)$, its p-isomorphic degree. For any class of languages \mathcal{C} , we refer to \leq_m^p -complete [\leq_m^p -hard] sets for \mathcal{C} as \mathcal{C} -complete [\mathcal{C} -hard] sets. Classes P, NP^k ($k > 0$), NP, EXP, NEXP are defined in the usual way. SAT is the standard NP-complete set

of satisfiable boolean formulae. The following TM index sets present classes P , NP and NP^k ($k > 0$) respectively.

$$\begin{aligned} PM &\stackrel{\text{def}}{=} \{i \mid M_i \text{ a DTM and } (\forall x) [T_{M_i}(x) \leq p_i(|x|)]\} \\ NPM &\stackrel{\text{def}}{=} \{i \mid M_i \text{ a NDTM and } (\forall x) [T_{M_i}(x) \leq p_i(|x|)]\} \\ NPM^k &\stackrel{\text{def}}{=} \{i \mid M_i \text{ a NDTM and } (\forall x) [T_{M_i}(x) \leq |i| \cdot |x|^k + |i|]\}, \text{ for any } k > 0 \end{aligned}$$

Productive sets over a language class \mathcal{C} are those sets that have a function witnessing the fact that the set does not belong to the class \mathcal{C} . Creative sets are defined to be the complement of productive sets. As we have mentioned earlier, the complexity of the creative set depends on the indexing of the language class. So, we use the index set, instead of the language class, to define the creativeness and productiveness.

Definition 2.1 For a given class of functions F and a set of TM indices I , language A is (F, I) -productive if there is a function f , $f \in F$ such that for every $i \in I$, $f(i) \in W_i$ iff $f(i) \in \bar{A}$. f is called (F, I) -productive function for A .

Definition 2.2 Set A is (F, I) -creative if A is r.e. and \bar{A} is (F, I) -productive.

When the meaning is clear by the context, we will refer to (F, I) -productive functions simply as productive functions. When F is the class of polynomial time functions, we will write (p, I) -creative for (F, I) -creative.

The standard definitions of creativeness for some classes in our notation will be as follows.

For the class r.e., the standard definition of creativeness [13] will be denoted in our notation as (rec, N) -creativeness, where rec is the class of (not necessarily total) recursive functions and N is the set of natural numbers.

For the classes EXP and $NEXP$, the existing creative sets [14] in our notations will be written as (p, PM) -creative and (p, NPM) -creative respectively.

For NP, k -creative sets [10] for $k > 0$ are denoted as (p, NPM^k) -creative.

NOTE : What we are calling creative and productive is referred as *completely creative* and *completely productive* in the terminology of [13] and [14]. According to [13], set A is (F, I) -*productive* if there is a function $f \in F$ such that for every $i \in I$, $W_i \subseteq A \Rightarrow f(i) \in A - W_i$. Creative sets are, as usual, complement of the productive sets. For the classes EXP, NEXP and r.e., the creativeness and completely creativeness are known to be equivalent (see [13] and [14]). For NP, as far as k -creativeness is concerned, the question of their equivalence is open. As for our definition, these are different if $P \neq NP$ (Next note).

For the definitions below, we will require the class of languages CONST, recognized by TMs working in *constant time*.

$$\text{CONST} \stackrel{\text{def}}{=} \{W_i \mid (\exists c)(\forall x) [T_{M_i}(x) \leq c]\}$$

One can define several sets of TM indices each presenting the class of languages CONST. We will be making use of mainly the following sets presenting CONST.

$$CM_{NP(r)} \stackrel{\text{def}}{=} \{i \mid M_i \text{ a NDTM and } (\forall x) [T_{M_i}(x) \leq |i|^{r+1}]\} \text{ for } r \geq 0$$

The set that we will most frequently use from the above sequence of sets is $CM_{NP(0)}$. We make the zero implicit and write it as simply CM_{NP} . The following proposition gives some properties of the class CONST.

Proposition 2.3 (i) CONST is the smallest class of languages containing finite sets and closed under union, complement and right concatenation with Σ^* .

(ii) CONST is a strict subclass of regular languages.

2.3 A new definition of creativeness for NP

We propose, in this section, a different definition of creativeness for NP, more general than k -creativeness.

For the language class NP, we use (p, CM_{NP}) -creative as the definition for creativeness.

2.3.1 (p, CM_{NP}) -creative sets are NP-hard

At first glance, (p, CM_{NP}) -creativeness may appear uninteresting since the (p, CM_{NP}) -productive sets, by construction, diagonalize only over the class CONST which is a very trivial class. However, it is the chosen indexing of the language class over which creativeness is defined that makes the creative set difficult or easy, not the language class itself. In fact, using a different indexing of the class CONST (see section 3.3), it can be shown that creative sets over this indexing are r.e.-complete ! The reason for this seemingly strange behavior is not too difficult to see. It will be made clear by the following theorem which shows that (p, CM_{NP}) -creative sets are NP-hard.

NOTE : If we use the definition of creativeness (and productiveness) given in the previous note, then some very trivial sets in P can be shown (p, CM_{NP}) -productive. An example is the set $TALLY = \{1^n \mid n \geq 1\}$, which can be shown to be (p, CM_{NP}) -productive with productive function $h = \lambda i. 1^{|i|+1}$ — Given any index $i \in CM_{NP}$, it is easy to see that either W_i is not a subset of $TALLY$ or else, W_i contains strings of size at most $|i|$. We need not bother about the first case and if the second case is true then $h(i) \in TALLY - W_i$.

Theorem 2.4 *Let the set A be (p, CM_{NP}) -creative. Then A is NP-hard.*

Proof. For $B \in NP$, let M_B be the NDTM recognizing B with running time bounded by $p_i(n)$ for some polynomial p_i . Define NDTM $M_{g(x)}$ as —

On input z , run M_B on x for at most $p_i(|x|)$ steps and accept if M_B accepts x .

Note that the function g can be computed in polynomial time and that $M_{g(x)}$ will work *independent of all inputs*. It is clear by construction that, $T_{M_{g(x)}}(z) \leq p_k(|x|)$ for some polynomial p_k .

Let $q(x) = g(x)0^{p_k(|x|)}$. From the paddability property of the indexing scheme we have that,

$$\phi_{q(x)} = \phi_{g(x)} \text{ and } (\forall z) [T_{M_{q(x)}}(z) = T_{M_{g(x)}}(z) \leq |q(x)|]$$

Therefore, $q(x) \in CM_{NP}$. Further,

$$W_{q(x)} = \begin{cases} \emptyset & \text{if } x \notin B \\ \Sigma^* & \text{if } x \in B \end{cases} \quad (2.1)$$

Let h be the productive function for A . Since $q(x) \in CM_{NP}$ we have, by definition

$$h(q(x)) \in A \Leftrightarrow h(q(x)) \in W_{q(x)} \quad (2.2)$$

Now using (2.1) and (2.2) we get, $x \in B \Leftrightarrow h(q(x)) \in W_{q(x)} \Leftrightarrow h(q(x)) \in A$. Therefore, $f \stackrel{\text{def}}{=} \lambda x. h(q(x))$ reduces B to A . \square

A discussion of the above proof : To reduce any set in NP to the given (p, CM_{NP}) -creative set, the above proof constructs a TM index for each instance of the NP set and then uses productive function on these indices to yield a reduction. TMs coded by these indices work independent of the input and therefore these TMs need work only for a constant time. This is the reason why the creativeness over such trivial

class of languages yields NP-hard sets. Moreover, it is easy to see that if we consider the set of TMs whose runtime is bounded by, say, an exponential in the length of the index (instead of the linear bound used in CM_{NP}), the corresponding creative set can be shown NEXP-hard. Thus, such a variation will vary the complexity of the corresponding creative set, while the language class over which creativeness is defined remains the same, viz., CONST.

The following lemma shows the invariance of (p, CM_{NP}) -creative sets under a polynomial increase in the bound on the runtime of TMs chosen to present the class CONST.

Lemma 2.5 *For any $r \geq 1$, A is (p, CM_{NP}) -creative iff A is $(p, CM_{NP(r)})$ -creative.*

Proof. (\Leftarrow) trivial.

(\Rightarrow) Let A be (p, CM_{NP}) -creative with productive function h . Define function h_r as — $h_r \stackrel{\text{def}}{=} \lambda i. h(i0^{|i|^{r+1}-|i|})$. Now, by the paddability property of the indexing scheme, for every $i \in CM_{NP(r)}$, $i0^{|i|^{r+1}-|i|} \in CM_{NP}$. So, for every $i \in CM_{NP(r)}$, $h_r(i) \in A$ iff $h(i0^{|i|^{r+1}-|i|}) \in A$ iff $h(i0^{|i|^{r+1}-|i|}) \in W_i$ iff $h_r(i) \in W_i$. \square

There exist (p, CM_{NP}) -creative sets in NP. Following ‘universal’ set is the most fundamental (p, CM_{NP}) -creative set.

$$K_{NP} \stackrel{\text{def}}{=} \{i \mid M_i \text{ a NDTM and accepts } i \text{ in at most } |i| \text{ steps}\}$$

For this set we note that,

Proposition 2.6 *$K_{NP} \in \text{NP}$ and is (p, CM_{NP}) -creative with identity productive function.*

Proof. Direct from the definitions. \square

2.3.2 k -Creative sets and (p, CM_{NP}) -creative sets

In this subsection, we compare our notion of creativeness with k -creativity. A k -creative set is trivially (p, CM_{NP}) -creative (recall that a k -creative set is denoted as (p, NPM^k) -creative in our notation), however, the converse does not appear to be true. To prove a (p, CM_{NP}) -creative set k -creative for some $k > 0$, one seems to require constraints on the length of productive function. The following theorem is the best that we could obtain.

Theorem 2.7 *If A is $(p, CM_{NP(r)})$ -creative with productive function h such that for some $t \leq r$, $|h(x)| = O(|x|^t)$, then A is $\lfloor r/t \rfloor$ -creative.*

Proof. Let $k = \lfloor r/t \rfloor$. Define TM $M_{g(i)}$ as —

On input x , accept iff M_i accepts x in $|i| \cdot |i|^{2 \cdot t \cdot k + 1/2} + |i|$ steps.

Function g is so chosen that $|i|^2 \leq |g(i)| = O[|i|^2]$. Clearly g is a polynomial time function and $T_{M_{g(i)}}(x) \leq O[|i|^{2 \cdot t \cdot k + 3/2}] \leq |i|^{2 \cdot t \cdot k + 2} \text{ (a.e. } i) \leq |g(i)|^{t \cdot k + 1} \text{ (a.e. } i)$. Therefore, $g(i) \in CM_{NP(r)} \text{ (a.e. } i)$. Further, $|h(g(i))| \leq O[|i|^{2 \cdot t}] \leq |i|^{2 \cdot t + 1/2 \cdot k} \text{ (a.e. } i)$ and therefore, for almost all $i \in NPM^k$, TM $M_{g(i)}$ accepts $h(g(i))$ iff M_i accepts $h(g(i))$. Therefore, we have that for every $i \in NPM^k$ and $i > i_0$ for some large enough i_0 ,

$$h(g(i)) \in W_i \Leftrightarrow h(g(i)) \in W_{g(i)} \Leftrightarrow h(g(i)) \in A$$

Thus $h' \stackrel{\text{def}}{=} h \circ g$ is a productive function for all $i > i_0$ and this can be easily modified to yield a total productive function, e.g., $h'' \stackrel{\text{def}}{=} \lambda i. h'(Pad(i_0, i))$ will be the total (p, NPM^k) -productive function. \square

The above result is in the same vein as the one proved by Homer [7]. It appears that to show a $(p, CM_{NP(r)})$ -creative set k -creative, one must have severe length restrictions on the productive function of the set. We have no evidence that such restricted productive functions will exist in general.

2.4 A new reducibility, preserving (p, CM_{NP}) -creativity

One can not immediately hope to prove that all NP-complete sets are (p, CM_{NP}) -creative, as it implies $P \neq NP$ (this follows from the fact that the sets in the class $CONST$ can not be (p, CM_{NP}) -creative). The goal in this section is to develop a new reducibility, called \leq_m^{pr} -reducibility, such that every \leq_m^{pr} -complete set for NP will be (p, CM_{NP}) -creative. (Next section will show that this reducibility is general enough to capture most NP-complete sets).

We begin with looking at a standard method (see, e.g., [13], [14]) to prove creativity for all languages in a given degree and see why does it not work for the NP-complete degree.

Let A be (p, CM_{NP}) -creative with productive function h and $A \leq_m^p B$ via f . Given any i , construct TM $M_{g(i)}$ as — On input x , compute first $|i|$ bits of $f(x)$ and accept iff M_i accepts these $|i|$ bits in $|i|$ steps.

Now, suppose $i \in CM_{NP}$. Then we have that M_i works only for $|i|$ steps on every input and since $M_{g(i)}$ provides it with the first $|i|$ bits of $f(x)$, M_i will accept these $|i|$ bits iff it accepts $f(x)$. So, for $x = h(g(i))$, $h(g(i)) \in W_{g(i)}$ iff $f(h(g(i))) \in W_i$. If we can ensure that $g(i) \in CM_{NP}$ then by productiveness, we will have $h(g(i)) \in W_{g(i)}$ iff $h(g(i)) \in A$ and by reduction, $h(g(i)) \in A$ iff $f(h(g(i))) \in B$. Combining these, we get, $f(h(g(i))) \in W_i$ iff $f(h(g(i))) \in B$ and therefore, $f \circ h \circ g$ will be the required productive function for B , showing B to be (p, CM_{NP}) -creative. But to have $g(i) \in CM_{NP}$, $M_{g(i)}$ must be able to compute first $|i|$ bits of $f(x)$ for $x = h(g(i))$ in at most $|g(i)|$ steps. This again imposes a length restriction (and a time restriction as well) on f and/or h , which, as we have said, is not likely to hold in general.

The above discussion suggests that if we use a restricted class of reductions instead of the whole polynomial class, then the above method may work. A close examination of the method reveals that the TM $M_{g(i)}$ needs to generate only the first $|i|$ bits of $f(g(i))$. If this can be done quickly without computing whole of $f(g(i))$ then the proof will go through. This motivates the following definition.

Definition 2.8 Function f is a *polynomial in range function* if there is a transducer M_a , computing f , which requires, along with the input x , $|x|$ written in binary on a work tape at the beginning of the computation and satisfies the following conditions :

1. $(\forall x) [T_{M_a}(x) \leq p_a(|x|)]$, and
2. $(\forall x) (\forall t \leq p_a(|f(x)|)) [| \phi_a^t(x) | + \log |x| \geq p_a^{-1}(t)]$

We shall refer to the functions defined above as pr-functions. Let PRF be the class of all pr-functions.

The idea behind pr-functions is that the transducers computing them will have to output bits regularly, i.e., if t steps of computation are over then the number of bits written on the output tape must be at least $p^{-1}(t)$ for some polynomial p . Such a transducer will have to output a large number of bits *without even scanning the whole input*. It is for this reason that they are provided with the length of the input at the beginning of computation. A transducer computing a pr-function need not even scan the whole input, however, in this paper, whenever we refer to pr-functions, we will mean pr-functions computed by transducers that scan the whole input.

We say that function f is *length-computable* if $(\forall x)(\forall y) [|x| = |y| \Rightarrow |f(x)| = |f(y)|]$ and its length function $length_f = \lambda n. f(1^n)$ is computable in polynomial time. The following lemma lists some basic properties of pr-functions.

Lemma 2.9 (i) *If f is a pr-function then f is honest.*

- (ii) f is a pr-function iff f is computable by a polynomial time transducer M_a satisfying the following property — $(\forall x)(\forall n \leq p_a^{-1}(|x|))$ first n bits of $f(x)$ are computable in at most $p_a(n + \log |x|)$ steps.
- (iii) Let the function $h = f \circ g$ be the composition of pr-functions f and g with the function g being length-computable, then h is also a pr-function.

Proof sketch.

- (i) Let M_a compute f . Then we have,

$$(\forall x)(\forall t \leq p_a(|f(x)|)) \left[|\phi_a^t(x)| + \log |x| \geq p_a^{-1}(t) \right]$$

Now, $T_{M_a}(x) \geq |x|$ implies $|\phi_a(x)| \geq |\phi_a^{|x|}(x)| \geq p_a^{-1}(|x|) - \log |x|$. Therefore f is honest.

- (ii) direct.

- (iii) Let the transducers M_a and M_b compute functions f and g respectively. Define the transducer M_c as —

On input x , using $|x|$ written on the work tape, compute $l = \text{length}_g(|x|)$ (since g is length-computable, $l = |g(x)|$ and the computation will require only a polynomial in $\log |x|$ time). Now compute g and f in parallel, i.e., compute first $\log |x|$ bits of $g(x)$; compute f on these bits; if and when it requires more bits, compute $\log |x|$ more bits of $g(x)$ and continue computation of f ; ... and so on.

M_c clearly is a polynomial time TM and the first n bits of $h(x)$ are generated in t steps with $t \leq O \left[\{p_a(\log |x| + n) + p_b(\log |x| + p_a(\log |x| + n))\}^2 \right] \leq$ a polynomial in $(\log |x| + n)$. Therefore, h is a pr-function. \square

Say $A \leq_m^{pr} B$ if there is a pr-function f (called pr-reduction) reducing A to B . The following theorem gives elementary facts about pr-reductions.

Theorem 2.10 (i) $A \leq_m^{pr} B \Rightarrow A \leq_m^p B$.

(ii) K_{NP} is \leq_m^{pr} -complete for NP.

Proof sketch.

(i) direct.

(ii) For any $W_i \in \text{NP}$, obtain a polynomial time reduction, $f = \lambda x. q(x)$, of W_i to K_{NP} as in Theorem 2.4, using (p, CM_{NP}) -creativity of K_{NP} and then define $\tilde{f} = \lambda x. 0^{|x|}q(x)$. \tilde{f} will be a pr-function by Lemma 2.9 and it will still be a reduction since $\tilde{f}(x)$ encodes the same TM as $f(x)$. \square

For any set A , let $d_{pr}(A)$ denote its \leq_m^{pr} -degree. We will write (pr, CM_{NP}) -creative for sets that are (p, CM_{NP}) -creative with a pr-function as its productive function. Now we give the theorem which was the motivation for defining pr-functions.

Theorem 2.11 A is (pr, CM_{NP}) -creative iff A is \leq_m^{pr} -hard for NP.

Proof. (\Rightarrow) The proof proceeds like the proof of Theorem 2.4. In that proof, with $B \in \text{NP}$ and $W_B = B$, the reduction of B to A is given by the function $f = \lambda x. h(q(x))$ where h is the productive function for A . We only have to ensure that f is a pr-function given that h is a pr-function. We will make the function q a length-computable pr-function and this would, by Lemma 2.9, make f a pr-function. By the construction of q , we know that for any x , $|q(x)|$ is bounded by $2 \cdot p_k(|x|)$. Define $e(x) = 0^{3 \cdot p_k(|x|) - |q(x)|}q(x)$. Since first $|x|$ bits of $e(x)$ are all zeroes it is, by Lemma 2.9, a pr-function and since for all x , $|e(x)| = 3 \cdot p_k(|x|)$, it is length-computable as well. We have, by the paddability

property of the indexing scheme, that $\phi_{e(x)} = \phi_{q(x)}$ and $T_{M_{e(x)}}(z) = T_{M_{q(x)}}(z) \leq |e(x)|$. Therefore, $\tilde{f} = \lambda x. h(e(x))$ will be the required pr-reduction.

(\Leftarrow) The proof will proceed along the lines of the method outlined at the beginning of the section. Let $K_{NP} \leq_m^{pr} A$ via f . Let M_a be the transducer computing f . We note that the transducer M_a , to compute $f(x)$, will also require $|x|$ written in binary on a work tape. Define TM $M_{g(i,k)}$ as —

On input x , compute $length = |i|^k$, run M_a on x , with $length$ as the length of x written on a work tape, for $p_a(length + |i|)$ steps. Accept iff M_i accepts the output of the above computation in $|i|$ steps.

Computation of $length$ requires at most $O[k^2 \cdot \log |i|]$ steps. So,

$$T_{M_{g(i,k)}}(x) \leq O[p_a(k^2 \cdot \log |i| + |i|)] \leq (k^2 \cdot |i|)^{|i|+1} \text{ (a.e. } i)$$

Choose g such that $((|a| + 2)^2 \cdot |i|)^{|i|+1} \leq |g(i, k)| = |i|^{|a|+2}$ (a.e. i) with the first $|i|$ bits all zeroes. Clearly, such a g can be chosen and by Lemma 2.9, it will be a pr-function.

Let $q = \lambda i. g(i, |a| + 2)$. We have, by the above calculations, $q(i) \in CM_{NP}$ (a.e. i). Now, TM $M_{q(i)}$ on input x will compute $length = |i|^{|a|+2}$ which is the length of $q(i)$ for almost all i . So, for almost all i , $length$ will be a correct guess of the length of x whenever $x = q(i)$ and therefore the computation of M_a will proceed correctly on these inputs. Thus we get that for almost all $i \in CM_{NP}$, $q(i) \in W_{q(i)}$ iff $f(q(i)) \in W_i$. We already have that for every i , $f(q(i)) \in A$ iff $q(i) \in K_{NP}$ iff $q(i) \in W_{q(i)}$. Combining the two, we get $f(q(i)) \in A$ iff $f(q(i)) \in W_i$ (a.e. i). Moreover, function $h \stackrel{\text{def}}{=} f \circ q$ will be a pr-function on large enough inputs since q is length-computable on almost all inputs.

This gives the required productive function for large enough inputs and from this a productive function that is correct on all inputs can be easily obtained, as was done in the proof of the Theorem 2.7. \square

2.5 The \leq_m^{pr} -reducibility

In this section, we will investigate the \leq_m^{pr} -reducibility by first defining a padding function which captures some of its properties and then go on to show that \leq_m^{pr} - and \leq_m^p -reducibilities are different everywhere.

2.5.1 A new kind of padding function

The most important reason for the only if part of Theorem 2.11 to hold is that TM M_i runs only for $|i|$ steps on every input and so TM $M_{q(i)}$ needs to generate only the first $|i|$ bits of $f(q(i))$. This suggests that if an NP-hard language A has the following property :

Given an instance y and any string x , one can find in polynomial time an instance of the form xz such that $y \in A$ iff $xz \in A$.

then the language should be (pr, CM_{NP}) -creative. Because then, given any i , choose an x such that $|x| \geq |i|$ and y such that $y \in A$ iff M_i accepts x in $|i|$ steps, using the hardness of A . Now, $h(i) = xz$ will be the productive function since for every $i \in CM_{NP}$, $xz \in W_i$ iff $x \in W_i$ iff $y \in A$ iff $xz \in A$. We define a notion of paddability which is based on the idea explained above. This notion is a generalization of the above property.

For every set A , define

$$LP_A \stackrel{\text{def}}{=} \{y\#x \mid \text{for every } y \in \{0, 1, \#\}^* \text{ and } x \in A\}$$

Definition 2.12 A is *lpr-paddable* if there exists a pr-reduction lp_A of LP_A to A . Function lp_A is called the *lpr-padding function* of the set A .

The following lemma lists some elementary results about lpr-paddable sets.

Lemma 2.13 (i) $A \leq_m^{pr} LP_A$.

(ii) $LP_A \leq_m^p A$.

(iii) $A \leq_m^p B \Leftrightarrow LP_A \leq_m^{pr} LP_B$.

(iv) A is *lpr-paddable* $\Leftrightarrow (\forall B) [B \leq_m^p A \Rightarrow B \leq_m^{pr} A]$.

(v) A is *lpr-paddable* $\Leftrightarrow (\exists B \in \mathbf{d}_m(A)) [LP_B \leq_m^{pr} A]$.

Proof.

(i) Immediate.

(ii) Immediate.

(iii) (\Rightarrow) Let $A \leq_m^p B$ via f . Define $\tilde{f} \stackrel{\text{def}}{=} \lambda y \# x. 1^{|y \# x|} \# f(x)$. \tilde{f} is a pr-function and it reduces LP_A to LP_B .

(\Leftarrow) Let $LP_A \leq_m^{pr} LP_B$ via g . Define $\tilde{g} \stackrel{\text{def}}{=} \lambda x. z$, where $g(1 \# x) = y \# z$. \tilde{g} is the required reduction.

(iv) (\Rightarrow) Let $B \leq_m^p A$ via f with the length of f bounded by polynomial p_k . Define, $\tilde{f} \stackrel{\text{def}}{=} lp_A \circ g$ where $g = \lambda x. 1^{|x| + p_k(|x|) - |f(x)|} \# f(x)$. Since first $|x|$ bits $g(x)$ are all ones and $|g(x)| = |x| + p_k(|x|) + 1$, it is a length-computable pr-function. Therefore, \tilde{f} is also a pr-function.

(\Leftarrow) Immediate from (ii).

(v) (\Rightarrow) Immediate from (ii).

(\Leftarrow) Let $LP_B \leq_m^{pr} A$ via f and $LP_A \leq_m^p B$ via g with the length of g bounded by polynomial p_k . Define $lp_A \stackrel{\text{def}}{=} f \circ h$ where $h = \lambda y \# x. 1^{2 \cdot p_k(|y \# x|) - g(y \# x)} \# g(y \# x)$. As in the previous part, h can be shown to be a length-computable pr-function and therefore lp_A is a pr-function. \square

It is clear from the above Lemma that every \leq_m^p -degree contains a maximum \leq_m^{pr} -degree and this \leq_m^{pr} -degree is the degree containing all lpr-paddable sets of the \leq_m^p -degree. Thus lpr-paddability provides a nice way of exhibiting (pr, CM_{NP}) -creativity of NP-complete sets.

Theorem 2.14 *For every $A \in NP$ the following holds.*

A is (pr, CM_{NP}) -creative iff A is NP-complete and A is lpr-paddable.

Proof. Follows from Lemma 2.13 and Theorem 2.11. □

All natural NP-complete sets seem to possess lpr-padding functions. For example,

Theorem 2.15 *SAT is lpr-paddable.*

Proof Sketch. lp_{SAT} is defined as follows —

On $y\#x$, initially output $|y| + |x|$ trivial clauses, clause i having the form :

$v_i \vee \bar{v}_i$, v_i being the i^{th} variable and then output clauses of x .

lp_{SAT} is clearly a lpr-padding function. □

The set K_{NP} can be easily seen to be lpr-paddable as a direct consequence of the paddability property of the indexing scheme. Therefore, we have

Corollary 2.16 *SAT is (pr, CM_{NP}) -creative and therefore in $d_{pr}(K_{NP})$.*

For natural NP-complete sets (see [6] for a comprehensive list of such sets) the lpr-padding function seems to be directly obtainable from the normal padding function (as defined in [3]) of the set. And since all natural NP-complete sets are believed to be paddable, it is not unreasonable to assume that they will be lpr-paddable as well. Are all paddable NP-complete sets lpr-paddable ? The answer is no (see Corollary 2.29).

Can we say that \leq_m^{pr} -degrees containing lpr-paddable sets are contained in a single isomorphism degree? There is no apparent reason to believe so. Sets in such degrees have a kind of padding function but these functions need not be invertible. However, it is not too difficult to see that any \leq_m^{pr} -degree containing lpr-paddable sets falls in a single honest degree.

Theorem 2.17 *Let $A \leq_m^{pr} B$. Then $A \leq_{m,si}^{pr} B$ if either (1) B is lpr-paddable or (2) A is lpr-paddable with lp_A a length-computable pr-function.*

Proof Sketch. Let $A \leq_m^{pr} B$ via f with the time taken to compute f bounded by polynomial p_k .

(1) Let the lpr-padding function, lp_B , of B be computed by the transducer M_b . Define another reduction \tilde{f} of A to B , $\tilde{f} \stackrel{\text{def}}{=} \lambda x. lp_B \left(1^{p_r(2 \cdot |x|) - |f(x)|} \# f(x) \right)$ where $r = \max\{k, b\}$. \tilde{f} can be easily seen to be a pr-reduction and it will be almost everywhere size-increasing since TM M_b will output at least $p_b^{-1}(p_r(2 \cdot |x|)) - \log(p_r(2 \cdot |x|)) \geq |x|$ a.e. bits on input of length $p_r(2 \cdot |x|) + 1$. From this, a pr-reduction that is everywhere size-increasing can be easily obtained.

(2) Let lp_A be computed by transducer M_a . Define $\tilde{f} \stackrel{\text{def}}{=} \lambda x. f \left(lp_A \left(1^{p_a(|x| + p_k(2 \cdot |x|))} \# x \right) \right)$. \tilde{f} is the required size-increasing pr-reduction. \square

Corollary 2.18 *If A is \leq_m^{pr} -complete for NP then A is $\leq_{m,si}^{pr}$ -complete.*

As mentioned earlier, not all paddable NP-complete sets are lpr-paddable. Thus, they are not (pr, CM_{NP}) -creative as well. Can we say that all paddable NP-complete sets are (p, CM_{NP}) -creative? We have an interesting result regarding this question. We show that if all paddable NP-complete sets are (p, CM_{NP}) -creative then *all* NP-complete sets are (p, CM_{NP}) -creative.

Theorem 2.19 *If every paddable NP-complete set is (p, CM_{NP}) -creative then every NP-complete set is (p, CM_{NP}) -creative.*

Proof. Let A be any NP-complete set. Define the set $RP_A \stackrel{\text{def}}{=} \{x\#y \mid y \in \{0,1,\#\}^*, x \in \{0,1\}^* \text{ \& } x \in A\}$. (This set is the symmetrical counterpart of the set LP_A .) Set RP_A will be NP-complete as well as paddable. By our assumption, RP_A is (p, CM_{NP}) -creative. Let h be the productive function for RP_A . Define function $f \stackrel{\text{def}}{=} \lambda z. [z, \text{ if } z \in \{0,1\}^*; x, \text{ if } z = x\#y \text{ \& } x \in \{0,1\}^*]$. Function f is a polynomial time reduction of RP_A to A .

Define NDTM $M_{g(i)}$ as —

On input x , let $y = [\text{first } |i| \text{ bits of } f(x), \text{ if } |f(x)| > |i|; f(x), \text{ otherwise}]$.

Accept iff M_i accepts y in $|i|$ steps.

Note that the function f is such that TM $M_{g(i)}$ can obtain y in $O(|y|)$ steps. Therefore, we can ensure by suitable padding that $g(i) \in CM_{NP}$ for every i . We get then, $h(g(i)) \in W_{g(i)}$ iff $h(g(i)) \in RP_A$. We also have $z \in RP_A$ iff $f(z) \in A$ and by the definition of $M_{g(i)}$, $z \in W_{g(i)}$ iff $f(z) \in W_i$ for every $i \in CM_{NP}$. Combining the above three, we get for every $i \in CM_{NP}$, $f(h(g(i))) \in W_i$ iff $h(g(i)) \in W_{g(i)}$ iff $h(g(i)) \in RP_A$ iff $f(h(g(i))) \in A$. Thus, $\tilde{h} \stackrel{\text{def}}{=} f \circ h \circ g$ is the required productive function for A . \square

2.5.2 Pr-simple sets

We have seen that a large number of NP-complete sets are \leq_m^{pr} -complete also. Naturally, the question arises — are there sets that are NP-complete but not \leq_m^{pr} -complete? If so, then these sets can possibly be non-creative also. In this subsection, we prove a very general result : degrees $\mathbf{d}_m(A)$ and $\mathbf{d}_{pr}(A)$ are different for all A . We will borrow the concept of simple sets from the recursion theory in proving this. In recursion theory,

simple sets are used to study non-complete, non-recursive sets. They are defined as the complements of immune sets. Their definitions are [13],

Definition 2.20 Set A is *immune* if A is infinite and $(\forall i)[W_i \subseteq A \Rightarrow W_i \text{ is finite}]$.

Definition 2.21 Set A is *simple* if A is r.e. and \bar{A} is immune.

The definition is prompted by the fact that productive sets contain infinite r.e. subsets and therefore no immune set can be productive. Immune sets can be defined by using a different but equivalent definition also : A is immune if A is infinite and for all size-increasing recursive functions f , $(\exists x)[f(x) \in \bar{A}]$. This is the definition that we translate down to pr-functions.

Definition 2.22 Set A is *pr-immune* if A is infinite and for each size-increasing pr-function f , $(\exists x)[f(x) \in \bar{A}]$.

Definition 2.23 Set A is *pr-simple* if A is r.e. and \bar{A} is pr-immune.

Theorem 2.24 If A is pr-simple then A can neither be lpr-paddable nor be (pr, CM_{NP}) -creative.

Proof Sketch. If A is lpr-paddable then $f \stackrel{\text{def}}{=} \lambda x. lp_A(x\#a)$, where lp_A is the lpr-padding function of A and $a \in \bar{A}$, is a pr-reduction of \emptyset to A . By Theorem 2.17, it can be modified to a size-increasing pr-reduction, which implies that A is not pr-simple.

If A is (pr, CM_{NP}) -creative then, by Theorems 2.11 and 2.17, $\emptyset \leq_{m,si}^{pr} A$ (since \emptyset is lpr-paddable with a length-computable lpr-padding function). Therefore, A is not pr-simple. \square

It is easy to construct pr-simple sets in P. Define set,

$$GAP \stackrel{\text{def}}{=} \{x \mid (\exists n)[exp(2n) \leq |x| < exp(2n+1)]\}$$

where $exp(0) = 1$ and $exp(n+1) = 2^{exp(n)}$.

Proposition 2.25 *Both GAP and \overline{GAP} are pr-simple.*

Proof Sketch. There can not be any size-increasing polynomial time function whose range is a subset of GAP or \overline{GAP} . This follows from the fact that both these sets have ‘exponentially large gaps’. Therefore, both will be pr-simple. \square

Now we will show that all \leq_m^p -degrees have a pr-simple set. The proof here is similar to Post’s construction of a tt-complete, simple set [13]. Post had accomplished this by taking the union of two disjoint r.e. sets, one tt-complete, and the other a simple set. The latter was obtained by putting in it an element of each r.e. set. In the present case, of the two disjoint sets one is a set in the given \leq_m^p -degree and the other is a pr-simple set in P . The latter is obtained by putting in it one element from the range of each size-increasing pr-function. However, since we are required to keep the set in P , we will generate only a prefix of the range element which can be computed in a short time because of the property of pr-functions. To carry out such a construction, we need an effective enumeration of all pr-functions. The following proposition guarantees it.

Proposition 2.26 *PRF is recursively presentable.*

Proof Sketch. We assume that for any i and for every x , $|x|$ is written in binary on a work tape of M_i on input x . For every i , define transducer $M_{t(i)}$ as —

On input x , simulate M_i on the input as long as the following condition is satisfied — After t steps of simulation let n be the number of bits that have been output, then $n \leq p_i(|x|)$ and $t \leq p_i(\log |x| + n)$. If the condition fails then output $|x|$ zeroes and stop; otherwise stop whenever M_i stops.

Since p_i is a fully time-constructible function, $\phi_{t(i)}$ will be a pr-function when t is sufficiently padded up and if ϕ_i is a pr-function then $\phi_{t(i)} = \phi_i$. \square

Let r_0, r_1, r_2, \dots be an enumeration of all pr-functions, with $r_i = \phi_{t(i)}$.

Theorem 2.27 For every A , $d_m(A)$ contains a pr-simple set.

Proof. Define function $prefix \stackrel{\text{def}}{=} \lambda x, n. [\text{first } n \text{ bits of } x \text{ if } n \leq |x|; x1^{n-|x|} \text{ otherwise}]$.

Define sets,

$$\begin{aligned} PS &\stackrel{\text{def}}{=} \{i\#1^{2^{2^{|i|}}} \#x \mid x = prefix(r_i(1^{2^{2^{|i|}}}), 2^{|i|})\} \\ S &\stackrel{\text{def}}{=} \{z \mid (\exists i)[(1 \leq |i| \leq \log \log |z|) \ \& \ i\#1^{2^{2^{|i|}}} \#prefix(z, 2^{|i|}) \in PS]\} \end{aligned}$$

Claim 1 : PS is p-printable.

Proof. Given n , compute for all j , $|j| \leq \log \log n$, $prefix(r_j(1^{2^{2^{|j|}}}), 2^{|j|})$. There will be $\log n$ such computations, each requiring at most $p_{i(j)}(\log n + \log n) \leq (\log n)^{p_k(|j|)}$ for some polynomial p_k (recall that $p_i(n) = n^{|i|} + 2^{|i|} \leq (\log n)^{p_k(\log \log n)} \leq$ a polynomial in n , steps. Hence PS is p-printable. \square

Claim 2 : $S \in P$.

Proof. Follows from Claim 1. \square

Claim 3 : \bar{S} is infinite.

Proof. A pr-function r_i can only cause strings of length at least $2^{2^{|i|}}$ to belong to S . In fact, since a prefix of length $2^{|i|}$ is computed, for any $n \geq 2^{2^{|i|}}$, it can cause at most $2^{n-2^{|i|}}$ strings of length n to belong to S . Therefore, for every n , $2^{2^k} \leq n < 2^{2^{k+1}}$: $|\{S\}_{=n}| \leq \sum_{m=1}^k 2^m * 2^{n-2^m} \leq 7/8 * 2^n$. So, \bar{S} is infinite. \square

Claim 4 : S is pr-simple.

Proof. Assume it is not. Then, since \bar{S} is infinite, there exists a size-increasing pr-function f such that $range(f) \subseteq \bar{S}$. Let $f = r_m$. Consider $f(1^{2^{2^{|m|}}}) = y$ (say). Since f is size-increasing, $|y| \geq 2^{2^{|m|}}$, i.e., $|m| \leq \log \log |y|$ and therefore, by definition, $y \in S$.

Contradiction. \square

Theorem 2.27 For every A , $d_m(A)$ contains a pr-simple set.

Proof. Define function $prefix \stackrel{\text{def}}{=} \lambda x, n. [\text{first } n \text{ bits of } x \text{ if } n \leq |x|; x1^{n-|x|} \text{ otherwise}]$.

Define sets,

$$\begin{aligned} PS &\stackrel{\text{def}}{=} \{i \# 1^{2^{2^{|i|}}} \# x \mid x = prefix(r_i(1^{2^{2^{|i|}}}), 2^{|i|})\} \\ S &\stackrel{\text{def}}{=} \{z \mid (\exists i)[(1 \leq |i| \leq \log \log |z|) \& i \# 1^{2^{2^{|i|}}} \# prefix(z, 2^{|i|}) \in PS]\} \end{aligned}$$

Claim 1 : PS is p-printable.

Proof. Given n , compute for all j , $|j| \leq \log \log n$, $prefix(r_j(1^{2^{2^{|j|}}}), 2^{|j|})$. There will be $\log n$ such computations, each requiring at most $p_{i(j)}(\log n + \log n) \leq (\log n)^{p_k(|j|)}$ for some polynomial p_k (recall that $p_i(n) = n^{|i|} + 2^{|i|} \leq (\log n)^{p_k(\log \log n)} \leq$ a polynomial in n , steps. Hence PS is p-printable. \square

Claim 2 : $S \in P$.

Proof. Follows from Claim 1. \square

Claim 3 : \bar{S} is infinite.

Proof. A pr-function r_i can only cause strings of length at least $2^{2^{|i|}}$ to belong to S . In fact, since a prefix of length $2^{|i|}$ is computed, for any $n \geq 2^{2^{|i|}}$, it can cause at most $2^{n-2^{|i|}}$ strings of length n to belong to S . Therefore, for every n , $2^{2^k} \leq n < 2^{2^{k+1}}$: $|\{S\}_{=n}| \leq \sum_{m=1}^k 2^m * 2^{n-2^m} \leq 7/8 * 2^n$. So, \bar{S} is infinite. \square

Claim 4 : S is pr-simple.

Proof. Assume it is not. Then, since \bar{S} is infinite, there exists a size-increasing pr-function f such that $range(f) \subseteq \bar{S}$. Let $f = r_m$. Consider $f(1^{2^{2^{|m|}}}) = y$ (say). Since f is size-increasing, $|y| \geq 2^{2^{|m|}}$, i.e., $|m| \leq \log \log |y|$ and therefore, by definition, $y \in S$. Contradiction. \square

For any string z , $|z| > 1$, let $div(z) \stackrel{\text{def}}{=} \langle y, x \rangle$ where $z = yx$ and $|x| < |y| = 2^{2^k}$ for some $k \geq 0$. Clearly, div is a polynomial time computable function. Define set

$$ImnS \stackrel{\text{def}}{=} \{z \mid div(z) = \langle y, x \rangle \text{ and } y \in \bar{S}\}$$

Claim 5 : $ImnS \in P$ and $ImnS \subseteq \bar{S}$.

Proof. $ImnS$ is clearly in P . $ImnS \subseteq \bar{S}$ because, arguing as in Claim 3, if $z \in ImnS$ and $div(z) = \langle y, x \rangle$ with $|y| = 2^{2^k}$, then any string in S with y as a prefix will have to be at least $2^{2^{k+1}}$ bits long, and $|yx| < 2 \cdot 2^{2^k} \leq 2^{2^{k+1}}$. \square

Define function g , $g \stackrel{\text{def}}{=} \lambda x. y_0x$ such that y_0 is the smallest string satisfying the following conditions — $div(g(x)) = \langle y_0, x \rangle$, $y_0 \in \bar{S}$ and $|y_0| \leq \max\{3, |x|^2\}$. (Such a y_0 will always exist because $|\{S\}_{=n}| \leq 7/8 * 2^n$ and will be obtainable in polynomial time because of p-printability of PS).

Claim 6 : g is a one-to-one, invertible, polynomial time function such that $range(g) \subseteq ImnS$.

Proof. Directly follows from the definition. \square

For any set A , define set $Smp(A) \stackrel{\text{def}}{=} S \cup \{z \mid z \in ImnS \ \& \ x \in A \text{ where } div(z) = \langle y, x \rangle\}$. Define function

$$\tilde{g} \stackrel{\text{def}}{=} \lambda z. \begin{cases} x & \text{if } z \in ImnS, \text{ where } div(z) = \langle y, x \rangle \\ a & \text{if } z \in S, \text{ where } a \in A \\ b & \text{otherwise, where } b \notin A \end{cases}$$

$Smp(A) \in \mathbf{d}_m(A)$ since function g reduces A to $Smp(A)$ and \tilde{g} reduces $Smp(A)$ to A . $Smp(A)$ is pr-simple since $\overline{Smp(A)}$ is infinite and is a subset of a pr-immune set, therefore itself a pr-immune set. \square

Corollary 2.28 *For every A , $d_{pr}(A)$ is properly contained in $d_m(A)$.*

Proof. Follows from the above Theorem and Lemma 2.13. \square

Corollary 2.29 *There is a pr -simple set which is p -isomorphic to SAT.*

Proof. Set $Smp(SAT)$ is a pr -simple set and it will be p -isomorphic to SAT since g is a size-increasing, invertible function. \square

Thus we have obtained a class of sets that is not (pr, CM_{NP}) -creative. Can we say that these sets are not even (p, CM_{NP}) -creative? That would provide us an NP-complete set which is not (p, CM_{NP}) -creative. However, as the following theorem shows, $Smp(A)$ will be (p, CM_{NP}) -creative for every A which is NP-hard.

Theorem 2.30 *For every A , If A is NP-hard then $Smp(A)$ is (p, CM_{NP}) -creative.*

Proof. Let $K_{NP} \leq_m^p A$ via f . Define TM $M_{q(i,y)}$ as — On input x , accept iff M_i accepts y in $|i|$ steps. Choose q such that $T_{M_{q(i,y)}}(x) \leq |q(i,y)|$. So, $q(i,y) \in CM_{NP}$. Let the length of the function $f \circ q$ be bounded by polynomial p_r . Define the productive function h for $Smp(A)$ as —
 $h \stackrel{\text{def}}{=} \lambda i. y_0 f(q(i, y_1))$ where $y_0 \in \bar{S}$, $|y_0| = 2^{2^k}$ for some k , $p_r(2 \cdot |i|) < |y_0| \leq [p_r(2 \cdot |i|)]^2$, and $y_1 = \text{prefix}(y_0, |i|)$.

It is easy to see that h can be computed in polynomial time (using p -printability of PS) and $\text{range}(h) \subseteq \text{Im}nS$ (since $\text{div}(h(i)) = \langle y_0, f(q(i, y_1)) \rangle$ and $y_0 \in \bar{S}$). Now, for every $i \in CM_{NP}$, $h(i) \in W_i$ iff $y_0 f(q(i, y_1)) \in W_i$ iff $y_1 \in W_i$ iff $q(i, y_1) \in W_{q(i, y_1)}$ iff $q(i, y_1) \in K_{NP}$ iff $f(q(i, y_1)) \in A$. Since $\text{range}(h) \subseteq \text{Im}nS$, $y_0 f(q(i, y_1)) \in Smp(A)$ iff $f(q(i, y_1)) \in A$. And therefore, $h(i) \in W_i$ iff $h(i) \in Smp(A)$. \square

Chapter summary

We have defined in this chapter a new notion of creativeness for NP, called (p, CM_{NP}) -creativity. This notion turns out to be a generalization of the earlier definition of creativeness for NP, viz., k -creativity. We have seen that every lpr-paddable NP-complete set will be (p, CM_{NP}) -creative and natural complete sets seem to be indeed lpr-paddable. We have also defined a subclass of NP-complete sets, viz., the class $\{Smp(A) \mid A \text{ is NP-complete}\}$, no member of which is lpr-paddable. However, even the sets in this class turn out to be (p, CM_{NP}) -creative.

Thus, the results derived seem to indicate a possible characterization of NP-complete sets; that they are (p, CM_{NP}) -creative. However, a proof of this will be difficult as it immediately implies $P \neq NP$.

Finally, as noted in section 2.3, one can use our technique to obtain definitions of creativeness for other classes than NP as well. We consider some of these definitions in the next chapter.

Chapter 3

Further Results on Creative Sets

3.1 Introduction

In this chapter we further investigate our notion of creativeness. In recursion theory, the creativeness property has been used to prove a fundamental result about r.e.-complete degree, viz., that all r.e.-complete sets are recursively isomorphic. We consider the polynomial isomorphism of complete degrees of bounded complexity classes using our notion of creativeness.

For the class NP, we could not obtain any significant general result. However, for a restricted class of reductions, called 1-L reductions, we show that all complete sets under 1-L reductions for NP are p-isomorphic. The result can be easily generalized to the classes varying from DLOG to NEXP. We also consider productive sets for NP and their relationship with diagonalization.

For the classes PSPACE, EXP, NEXP and r.e., we obtain new definitions of creativeness by defining suitable indexing of the class CONST. For EXP, NEXP and r.e., we show that these definitions are equivalent to the earlier ones given in [14] and [13]. We go on to prove that creative sets in EXP and NEXP indeed characterize the

complete degrees of EXP and NEXP respectively. For the class PSPACE, we show that creative sets characterize logspace-complete sets and all logspace-complete sets are complete under one-one, size-increasing logspace reductions.

The chapter is organized as follows. In section 3.2, we investigate the properties of creative and productive sets for the class NP. In section 3.3 we obtain a new definition of creativeness for the classes EXP, NEXP and r.e. and describe some of their properties. Section 3.4 contains a definition of creativeness for the class PSPACE and associated results.

3.2 Creative sets in NP

In this section, we investigate some more questions concerning (p, CM_{NP}) -creative sets inspired by the properties of creative sets for r.e. class.

3.2.1 Polynomial isomorphism and (p, CM_{NP}) -creative sets

The isomorphism question for NP is : are all NP-complete sets p-isomorphic ? This has motivated a lot of research on NP-complete sets, yet the question remains far from settled. Berman and Hartmanis [3] were first to address the question and they conjectured that all NP-complete sets are p-isomorphic. However, later Joseph and Young [10] came up with k -creative sets that did not appear to be p-isomorphic to standard NP-complete problems and therefore they conjectured the converse.

In recursion theory, it is known that all r.e.-complete sets under many-one reductions are indeed isomorphic. We will try to translate the proof of this to the polynomial settings. The proof consists of three steps. First, one defines creative sets for the class and shows that all many-one complete sets are creative. Next, one proves that all creative sets are complete under one-one reductions and finally, that all one-one complete

sets are isomorphic. In the present context, we have seen that there is some evidence to believe that all NP-complete sets are (p, CM_{NP}) -creative. For a possible third step, one may consider the result by Berman and Hartmanis [3] that all sets complete under one-one, size-increasing and invertible polynomial time reductions are p-isomorphic. In that case, the second step will be to prove that all creative sets are complete under one-one, size-increasing and invertible polynomial time reductions. However, it is not at all clear how this step can be carried out. In fact, we could not prove this second step even with only one of the three conditions. The only general result that we have obtained says that all (p, CM_{NP}) -creative sets are complete under *exponentially honest* reductions.

We say that functions f is *exponentially honest* if there is a polynomial p such that $(\forall x)[p^{-1}(|f(x)|) \leq |x| \leq 2^{p(|f(x)|)}]$. This definition is slightly different from the one given by Ganesan and Homer [5], who define exponentially honest functions as satisfying the property $(\forall x)[p^{-1}(|f(x)|) \leq |x| \leq 2^{|f(x)|}]$.

Theorem 3.1 *Let $A \in NP$ and (p, CM_{NP}) -creative. Then A is complete for NP under exponentially honest reductions.*

Proof. Let $A \in NP^k$ for some $k > 0$. There will be a DTM M_A recognizing the set A in $O(2^{p_k(n)})$ time. Define NDTM $M_{g(x)}$ as — On input z , if $p_k(|z|) \geq \log |x|$ then accept z iff $x \in K_{NP}$ else accept z iff $z \notin A$ (using M_A).

It can be seen that $T_{M_{g(x)}}(z) \leq p_r(|x|)$ for some polynomial p_r . Choosing g such that $|g(x)| \geq p_r(|x|)$ for every x , we get that $g(x) \in CM_{NP}$ for every x and

$$W_{g(x)} = \begin{cases} \{\bar{A}\}_{<n} \cup \Sigma_{\geq n} & \text{if } x \in K_{NP} \\ \{\bar{A}\}_{<n} & \text{otherwise} \end{cases}$$

where $n = p_k^{-1}(\log |x|)$. Now, let h be the productive function for A . By creativeness we have, $h(g(x)) \in W_{g(x)}$ iff $h(g(x)) \in A$. If for some x , $|h(g(x))| < p_k^{-1}(\log |x|)$ then

$h(g(x)) \in A$ iff $h(g(x)) \in W_{g(x)}$ iff $h(g(x)) \in \bar{A}$, a contradiction. Therefore for every x , $|h(g(x))| \geq p_k^{-1}(\log |x|)$ and then we have that $x \in K_{NP}$ iff $h(g(x)) \in W_{g(x)}$ iff $h(g(x)) \in A$. Since K_{NP} is complete for NP under size-increasing reductions, it follows that A is complete for NP under exponentially honest reductions. \square

As the isomorphism question appears very difficult to answer, we address a simpler question by considering a stronger class of reductions than polynomial time many-one. We take the reductions to be 1-L reductions (defined below) and then show that all NP-complete sets under 1-L reductions are p-isomorphic.

1-L reductions are functions computable by logspace-bounded Turing machines which have a one-way input tape and begin their computation with $\lceil \log n \rceil$ cells marked off on a work tape. They were introduced in [9] for studying complete sets for DLOG. For a class \mathcal{C} , we say that a set is 1-L-complete for \mathcal{C} if the set is complete for \mathcal{C} under 1-L reductions. Allender [1] has shown that 1-L-complete sets for every class in the list (DLOG, NLOG, P, NP, PSPACE, DTIME($2^{O(1)}$), EXP, NTIME($2^{O(1)}$), NEXP) are complete under size-increasing and *strongly invertible* reductions, where a function f is *strongly invertible* if there is a DTM which, on input y , prints every element in $f^{-1}(y)$ in polynomial time. Note that a strongly invertible function need not be one-one, though it must be poly-one. He further shows that for the classes PSPACE and EXP, 1-L-complete sets must be complete under size-increasing, strongly invertible *and* one-one reductions as well; thus showing that all 1-L-complete sets are p-isomorphic for these two classes. Ganesan and Homer [5] have proved a similar result for the class NEXP. In the following we will prove a general result : for every class in the above list, 1-L-complete sets are p-isomorphic.

We use a slightly modified enumeration of TMs. This enables us to make the proof easier. Let M_0, M_1, \dots be the standard enumeration of TMs. We write TM indices of the above enumeration in binary. Let $\#$ be a symbol not in $\{0, 1\}$. Define TM

$M_{c(s_1\#s_2\#\dots\#s_k\#i)}$, where $s_1, \dots, s_k, i \in \{0, 1\}^*$, as —

On input z , simulate TM M_i on z . If M_i has at least two work tapes, at least three states and the state number 3 is neither the start state nor an accepting state then : whenever M_i goes to state 3, let r be the number written on the first work tape, write the r^{th} bit of each of the strings s_1, \dots, s_k on the second work tape (If $|s_m| < r$ for some m then its r^{th} bit is taken to be ϵ) and continue with simulation.

Both the functions c and c^{-1} can be computed in polynomial time.

NOTE : A simpler way of defining the above TM is by making it write s_1, \dots, s_k right at the beginning on a work tape and then begin the simulation of M_i with suitable modifications. However, for proving the desired isomorphism result for the classes DLOG and NLOG, we will require the TM $M_{c(s_1\#s_2\#\dots\#s_k\#i)}$ to work within $\log(|s_1| + \dots + |s_k| + |i|)$ space whenever TM M_i works within $\log |i|$ space. This requirement will not be satisfied by this simpler definition and therefore we have to use the more complicated definition.

Define the new enumeration of TMs as — M'_0, M'_1, \dots where we write the TM indices as strings over $\{0, 1, \#\}$. We shall use primed variables to vary over indices in this enumeration. If $j' = s_1\#s_2\#s_3\#\dots\#s_k\#i$, with each s_m and i in $\{0, 1\}^*$ for $1 \leq m \leq k$, then TM $M'_{j'} \stackrel{\text{def}}{=} M_{c(j')}$.

We shall prove the following theorem for NP, however, it can be easily be generalized to any class in the above given list of classes. First we give a different but equivalent

definition of creativeness for NP based on the new enumeration of TMs. Let

$$CM'_{NP} \stackrel{\text{def}}{=} \{j' \mid M'_{j'} \text{ a NDTM and } (\forall x) T_{M'_{j'}}(x) \leq |j'|\}$$

The set

$$K'_{NP} \stackrel{\text{def}}{=} \{j' \mid M'_{j'} \text{ a NDTM and accepts } j' \text{ in } |j'| \text{ steps}\}$$

is (p, CM'_{NP}) -creative with identity productive function. It is easy to see that a set is (p, CM'_{NP}) -creative iff it is (p, CM_{NP}) -creative.

Let M be a TM computing a 1-L reduction. A *configuration of M of size n* is a partial ID of M on input of size n . It is written as a 4-tuple containing the current input, output and work tape head positions and the contents of the work tape. Thus, the size of each such configuration will be bounded by a constant times $\log n$.

Theorem 3.2 *Let $K'_{NP} \leq_m^p A$ via f with f being a 1-L reduction. Then $K'_{NP} \leq_{si,1,i}^p A$.*

Proof. Define TM $M'_{s_1 \# s_2 \# s_3 \# t \# q(c(i'))}$ as —

On input x , check if $|s_1| = |s_2| = |s_3| = |i'|^2$. Reject if the above equalities do not hold. Otherwise if $s_1 = s_2$ then reject; if $s_1 \neq s_2$ and $s_1 = s_3$ then accept; otherwise accept iff $i' \in K'_{NP}$.

One can pad q sufficiently to ensure that for all i' , $|q(c(i'))| = p_k(|i'|)$ for some polynomial p_k and $s_1 \# s_2 \# s_3 \# t \# q(c(i')) \in CM'_{NP}$. Let the function f be bounded by the polynomial p and computed by the TM M . Define functions $gsize$, gap and $gapbd$ as : $gsize(n) \stackrel{\text{def}}{=} 3n^2 + 4 + p_k(n)$, $gap(n) \stackrel{\text{def}}{=} [p(gsize(gap(n-1)))]$ if $n > 0$; 0 if $n = 0$, and $gapbd(n) \stackrel{\text{def}}{=} gap(\mu_m\{n \leq gap(m)\})$. Function $gapbd$ is computable in polynomial time in n . Define a function g as given by the following procedure.

begin

Input i' .

1. Let $n = \text{gapbd}(|i'|)$ and inum be the position of the string i' in the lexicographic ordering of strings over $\{0, 1, \#\}$.
2. Create a labelled diagraph (possibly with multiple edges) $G = (V, E)$ with V being the set of all configurations of M on input of size $\text{gsize}(n) [= 3.n^2 + 4 + p_k(n)]$ and E containing one or two edges labelled $l, l \in \{0, 1, \epsilon\}$ from configuration C_m to configuration C_k iff M moves from configuration C_m to configuration C_k in a single step and consumes the input l . So, if there are multiple edges between two configurations, then they must have different labels. The label of a path in G will be the concatenation of the labels of its edges. The graph G can be converted in a directed acyclic graph as the existence of a cycle in the graph implies either that the TM M will never halt on some input or that the configurations in the cycle are never reached from the starting configuration. In either case, the cycle can be broken by deleting one edge of the cycle.
3. Let C_{init} be the starting configuration, $v(n)$ be the total number of vertices in G ($v(n)$ will be bounded by some polynomial in n) and \mathcal{C} be the set of all configurations reachable from C_{init} after having consumed exactly n^2 input bits.
4. For each $C \in \mathcal{C}$, compute the number of paths from C_{init} to C and let C_{max} be the configuration that has the maximum number of paths from C_{init} . (There will be at least $2^{n^2}/v(n)$ number of paths from C_{init} to C_{max} .)
5. Let $\text{path}_{\text{inum}}$ be the inum^{th} largest path, in the lexicographic ordering of the labels, from C_{init} to C_{max} (if inum is greater than number of paths from C_{init} to C_{max} then take the largest such path) and l_{inum} be its label. ($|l_{\text{inum}}| = n^2$.)

6. If $l_{inum} \neq 1^{n^2}$ then output $l_{inum} \# 1^{n^2} \# 1^{n^2} \# 1^{p_*(n) - |q(c(i'))|} \# q(c(i'))$
 else output $l_{inum} \# 0^{n^2} \# 0^{n^2} \# 1^{p_*(n) - |q(c(i'))|} \# q(c(i'))$.

end

Let $h \stackrel{\text{def}}{=} f \circ g$. The following lemmas complete the proof.

Lemma 3.3 *Function g is computable in polynomial time.*

Proof. Steps 1, 2, 3 and 6 can be easily carried out in polynomial time in n , which is not greater than a polynomial in $|i'|$. For steps 4 and 5, one requires to compute the number of paths between two given configurations. The number of paths between every pair of configurations can be computed in polynomial time by simple dynamic programming techniques as the graph G is a directed acyclic graph. Therefore, g is polynomial time computable. \square

Lemma 3.4 $K'_{NP} \leq_m^p A$ via h .

Proof. By the construction of g , one can easily see that $g(i') \in CM'_{NP}$ and $g(i') \in W'_{g(i')}$ [$W'_{g(i')}$ is the language accepted by TM $M'_{g(i')}$] iff $i' \in K'_{NP}$. By the creativeness of K'_{NP} , $g(i') \in W'_{g(i')}$ iff $g(i') \in K'_{NP}$. Thus, we get, $i' \in K'_{NP}$ iff $g(i') \in K'_{NP}$ iff $f(g(i')) \in A$. \square

Lemma 3.5 *For all n such that $2^{[gap(n)]^2} / v(gap(n)) \geq 3^{gap(n)+1}$, and for every i' and j' such that $gap(n-1) < |i'|, |j'| \leq gap(n)$, $h(i') \neq h(j')$ as well as $|h(i')|, |h(j')| \geq gap(n) + 1$.*

Proof. Consider any number n satisfying $2^{[gap(n)]^2} / v(gap(n)) \geq 3^{gap(n)+1}$ and let $m = gap(n)$. Then, for every i' satisfying $gap(n-1) < |i'| \leq m$, $|g(i')| = gsize(m)$ and

the TM M , on input $g(i')$, will be in the configuration C_{max} after consuming first m^2 symbols of $g(i')$. M will be in the configuration C_{max} for at least $2^{m^2}/v(m) \geq 3^{m+1}$ different assignments to the first m^2 symbol positions. Each one of this assignment will correspond to a *distinct* path from C_{init} to C_{max} . Suppose that for any two such paths, $path_1$ and $path_2$, the partial output of M at the configuration C_{max} is same. Now consider the output of M on strings $i'_1 \stackrel{\text{def}}{=} path_1 \# path_1 \# path_2 \# \# q(c(j'))$ and $i'_2 \stackrel{\text{def}}{=} path_2 \# path_1 \# path_2 \# \# q(c(j'))$ where j' is any TM index with $|j'| = m$. Clearly, $f(i'_1) = f(i'_2)$. By construction of q , we have that $i'_1 \notin W'_{i'_1}$ while $i'_2 \in W'_{i'_2}$. By the creativeness of K'_{NP} , we get, $i'_1 \notin K'_{NP}$ and $i'_2 \in K'_{NP}$, thus contradicting the fact that f is a reduction. Therefore, the partial output of M on each path from C_{init} to C_{max} will be different and since each such output is of the same length we conclude that the output of M will be different on all the strings of length $gsize(m)$ whose first m^2 symbols are different from each other and leave M in the configuration C_{max} . From this we can also conclude that M will have output at least $m + 1$ bits by the time it reaches C_{max} since otherwise there will be two paths having the same partial output. \square

Lemma 3.6 *Function h is one-one and size-increasing almost everywhere.*

Proof. The above lemma shows that h is size-increasing almost everywhere and for almost all n and every i' and j' , if $gap(n-1) < |i'|, |j'| \leq gap(n)$ then $h(i') \neq h(j')$. Suppose that there are infinitely many i' and j' such that $h(i') = h(j')$. Then there are numbers n and m , $n > m$ such that $gap(m-1) < |j'| \leq gap(m)$ and $gap(n-1) < |i'| \leq gap(n)$. We have, $|g(j')| = gsize(gap(m))$ and $|g(i')| = gsize(gap(n))$. Therefore, $|h(j')| \leq p(gsize(gap(m))) = gap(m+1) \leq gap(n)$, by the definition of gap . From the above lemma, we have, $|h(i')| \geq gap(n) + 1$, and therefore $|h(j')| < |h(i')|$, a contradiction. So h is one-one and size-increasing almost everywhere. \square

The function h can be easily modified to yield an everywhere size-increasing and one-one function \tilde{h} . Let $\text{preimage}(y) \stackrel{\text{def}}{=} \{x \mid f(x) = y \ \& \ |x| \leq |y|\}$.

Lemma 3.7 (proved in [1]) *For function f , $\text{preimage}(y)$ is computable in time polynomial in $(|y| + |\text{preimage}(y)|)$.*

Proof Sketch. Compute the configuration graph of the logspace-bounded NDTM M_f that on input y , guesses x with $|x| \leq |y|$ and accepts if $f(x) = y$. To enumerate elements of $\text{preimage}(y)$, enumerate all paths in the graph from the starting configuration to the accepting configuration. This can be done in polynomial time in the number of such paths. \square

Using the above lemma along with the fact that the function g is invertible in polynomial time, we conclude that \tilde{h} can also be inverted in polynomial time. \square

Corollary 3.8 *All 1-L complete sets for NP are p-isomorphic.*

Proof. The set K'_{NP} is complete under one-one, size-increasing and polynomial time invertible reductions for NP. In [3], it was shown that all sets that are reducible to each other by one-one, size-increasing and polynomial time invertible reductions are p-isomorphic. The corollary follows. \square

3.2.2 Productive sets and diagonalization

One of the fundamental properties of creative sets for r.e. class is that their complements, i.e., the productive sets, diagonalize over the class of r.e. sets with the productive functions as *witnesses* : given any index i , the productive function maps it to the symmetric difference of the productive set and W_i , thus showing the productive set to be different from W_i . Let us first formalize these notions.

For any language class \mathcal{C} presented by a TM index set I , for any language A and for any recursive function f , we say that f *witnesses* A *outside* \mathcal{C} if

$$(\forall i)[(i \in I) \Rightarrow (\exists x)[f(x) \in L \Leftrightarrow f(x) \notin W_i]]$$

And if there is a function witnessing A outside \mathcal{C} then we say that A *diagonalizes over* \mathcal{C} . For any (F, I) -productive set one immediately obtains,

Proposition 3.9 *If A is (F, I) -productive with productive function f then f witnesses A outside $\mathcal{L}(I)$.*

Does such a diagonalization property hold for bounded complexity classes as well? Here, we try to answer this question for the class NP. This property, directly translated down to NP reads : there are creative sets in NP with polynomial time productive functions, such that their complements diagonalize over NP with the productive functions as witnesses. This property, if true, immediately implies $\text{NP} \neq \text{co-NP}$ and therefore proving it true would be very difficult, if at all possible. So, we restrict ourselves to answering the question *assuming* $\text{NP} \neq \text{co-NP}$. Even this question seems very difficult to answer because of the following reasons : when we take a creative set in NP, e.g., k -creative or (p, CM_{NP}) -creative, then its complement, by the above Proposition, diagonalizes only over the class NP^k or CONST respectively. And when we consider (p, I) -creative sets with I a usual TM presentation of NP, they do not belong to NP, e.g., (p, NPM) -creative sets are NEXP-hard (see next section). This, however, leaves open the possibility that some unusual presentation I of NP exists for which (p, I) -creative sets *will* belong to NP.

If we further weaken the question by dropping the condition that the productive functions should be computable in polynomial time then it can be answered positively. The following result is due to Kozen [12] who had proved it for every subrecursive class, we reproduce it only for the class NP.

Theorem 3.10 *Let I be a TM presentation of the class NP. For every recursive set A , $A \notin \text{NP}$ iff A is (rec, I) -productive.*

Proof. (\Leftarrow) trivial.

(\Rightarrow) Define TM M_a as follows —

On input i , output the smallest number n for which we have $n \in \bar{A}$ iff $n \in W_i$.

Let $f \stackrel{\text{def}}{=} \phi_a$. Since $A \notin \text{NP}$, A and all the languages in NP are recursive, for all indices $i \in I$, $f(i)$ will converge and produce the desired witness. \square

Corollary 3.11 *If $\text{NP} \neq \text{co-NP}$ then there are (rec, NPM) -creative sets in NP.*

Proof. By the above Theorem, every recursive set outside NP is (rec, NPM) -productive. Since $\text{NP} \neq \text{co-NP}$, all co-NP-hard sets will be (rec, NPM) -productive and therefore all NP-complete sets will be (rec, NPM) -creative. \square

We have some results for polynomial time productive functions as well. Our first result shows that if there are (p, I) -creative sets in NP with I a TM presentation of NP then either the productive functions for these sets must be dishonest or the presentation I is unnatural. Next, we show that at least *some* productive functions of $(p, \text{CM}_{\text{NP}})$ -productive sets do witness the set to be outside NP.

Creative sets over NP

In this section, whenever we refer to a (p, I) -creative sets, it will be assumed that I is some TM presentation of NP.

We first give a technical lemma and then prove our main result. Define

$$\text{NPM}_\alpha \stackrel{\text{def}}{=} \{i \mid M_i \text{ is an NDTM and } (\forall z) T_{M_i}(z) \leq |z|^{\alpha(i)}\}$$

Let $t_\alpha \stackrel{\text{def}}{=} \lambda x. |x|^{\alpha(x)}$. Define

$$C_\alpha \stackrel{\text{def}}{=} \{W_i \mid (\forall x) T_{M_i}(x) \leq O[t_\alpha(x)]\}$$

Lemma 3.12 *For every α , with $\inf_{i \rightarrow \infty} \alpha(i) \rightarrow \infty$ and t_α fully time-constructible, $\text{NP} \subset C_\alpha$.*

Proof. The class C_α can be easily shown to contain sets that diagonalize over NP as for every set in NP, the function t_α is a constant. \square

Theorem 3.13 *For every α satisfying (1) α is a monotonic non-decreasing function a.e., (2) $\inf_{i \rightarrow \infty} \alpha(i) \rightarrow \infty$ and (3) t_α is fully time-constructible : if A is (p, NPM_α) -creative with an honest productive function then $A \notin \text{NP}$.*

Proof. We show that A is C_β -hard for some β with $\inf_{i \rightarrow \infty} \beta(i) \rightarrow \infty$ and t_β fully time-constructible. The proof is along the same lines as that of Theorem 2.4.

Let $B \in C_\beta$ with TM M_B recognizing it in $O[t_\beta(n)]$ time. Define TM $M_{g(x)}$ as — on input z , reject if $|z| \leq |x|$, otherwise run TM M_B on x and accept iff it accepts x . Clearly, $T_{M_{g(x)}}(z) \leq [|z|, \text{ if } (|z| \leq |x|); O[|x|^{\beta(x)}], \text{ otherwise}] \leq |z|^{c \cdot \beta(x)}$ for some constant c . Let h be the honest productive function for A with $p_r^{-1}(|i|) \leq |h(i)| \leq p_r(|i|)$. Choose g such that it is computable in polynomial time and for every x , $|g(x)| > p_r(|x|)$. Also let $\beta = \lambda x. \alpha(x)/c$. So, $T_{M_{g(x)}}(z) \leq |z|^{\alpha(x)} \leq |z|^{\alpha(g(x))}$ (a.e. x) (since α is monotonically non-decreasing a.e.). Thus, for every $x > x_0$ for some fixed x_0 , $g(x) \in \text{NPM}_\alpha$ and $h(g(x)) > |x|$. Now, by construction, we have,

$$W_{g(x)} = \begin{cases} \emptyset & \text{if } x \notin B \\ \Sigma_{>|x|} & \text{if } x \in B \end{cases}$$

So, for every $x > x_0$, $x \notin B \Rightarrow h(g(x)) \notin W_{g(x)} \Rightarrow h(g(x)) \notin A$ and $x \in B \Rightarrow h(g(x)) \in W_{g(x)}$ (since $|h(g(x))| > |x| \Rightarrow h(g(x)) \in A$). Thus, function $h \circ g$ reduces B to A almost everywhere which can be easily modified to a reduction.

Therefore, A is C_β -hard. Since $\beta = \alpha/c$, $\inf_{i \rightarrow \infty} \beta(i) \rightarrow \infty$ and t_β is fully time-constructible it follows, from Lemma 3.12, that $A \notin \text{NP}$. \square

One feels that any natural presentation of NP will be some NPM_α where α satisfies the conditions in the statement of the above theorem. We have seen that in that case every polynomial time productive function of a (p, NPM_α) -creative set in NP must be dishonest.

$(p, \text{CM}_{\text{NP}})$ -creative sets

In this section, we show that if a $(p, \text{CM}_{\text{NP}})$ -productive set is not in NP then this fact is witnessed by a productive function of that set. Note that this property is weaker than the one desired which requires that *every* productive function for the set should be a witness.

Theorem 3.14 *Let the class NP be presented by NPM. Then for any $(p, \text{CM}_{\text{NP}})$ -productive set A , $A \notin \text{NP}$ iff there is a $(p, \text{CM}_{\text{NP}})$ -productive function for A witnessing A outside NP.*

Proof. (\Leftarrow) trivial.

(\Rightarrow) Let A be $(p, \text{CM}_{\text{NP}})$ -productive with productive function h . Since $A \notin \text{NP}$ we have, by Lemma 3.10, that A is (rec, NPM) -productive with productive function f (say). Define

$$\tilde{h} \stackrel{\text{def}}{=} \lambda i. \begin{cases} f(j) & \text{if } i = j0^n \text{ and computation of } f(j) \text{ halts within } n \text{ steps} \\ h(i) & \text{otherwise} \end{cases}$$

Clearly, \tilde{h} is a polynomial time function. $\tilde{h}(i)$ will be different from $h(i)$ only if $i = j0^n$ with $n \geq |f(j)|$. We know that if f converges on j then $f(j) \in W_j \Leftrightarrow f(j) \in \bar{A}$. Therefore, for these i we have,

$$\tilde{h}(i) \in W_i \Leftrightarrow f(j) \in W_{j0^n} \Leftrightarrow f(j) \in W_j \Leftrightarrow f(j) \in \bar{A} \Leftrightarrow \tilde{h}(i) \in \bar{A}$$

So, \tilde{h} is a (p, CM_{NP}) -productive function for A . Since f is a (rec, NPM) -productive function for A , for every $j \in NPM$, there will be a large enough n such that $\tilde{h}(j0^n) = f(j)$ and then $\tilde{h}(j0^n) \in \bar{A}$ iff $\tilde{h}(j0^n) \in W_j$. Thus, \tilde{h} witnesses A outside NP. \square

Corollary 3.15 *co-NP \neq NP iff for every (p, CM_{NP}) -productive set A , A has a productive function witnessing A outside NP.*

Proof. (\Rightarrow) Any (p, CM_{NP}) -productive set A will be co-NP-hard (from Theorem 2.4) so $A \notin \text{NP}$. Then use Theorem 3.14.

(\Leftarrow) Consider \bar{K}_{NP} , which is in co-NP. Since it diagonalizes over NP by hypothesis, we have the result. \square

3.3 Creative sets in EXP, NEXP and r.e.

Our method of defining creativeness, i.e., choosing a suitable TM presentation of the class CONST and giving the productive function over it, can be used to obtain the definitions of creativeness for classes higher than NP as well. In this section, we obtain the creativeness definitions for the classes EXP, NEXP and r.e. However, these definitions do not yield anything new, as they turn out to be equivalent to earlier ones. Nevertheless, we prove a new result for NEXP and EXP — Creative sets for these classes are \leq_m^p -hard for NEXP and EXP respectively. This answers some questions raised in [14], and combined with a result proved in [14] shows that creativeness and many-one hardness for the classes EXP and NEXP are equivalent notions, thus providing an alternative characterization for many-one complete sets of these classes.

As observed in section 2.3, the creativeness definitions for the higher classes can be obtained by simply modifying the the bound on the runtime of TMs presenting the

class CONST. Define,

$$\begin{aligned} CM_E &\stackrel{\text{def}}{=} \{i \mid M_i \text{ is a DTM and } (\forall x) [T_{M_i}(x) \leq 2^{|i|}]\} \\ CM_{NE} &\stackrel{\text{def}}{=} \{i \mid M_i \text{ is an NDTM and } (\forall x) [T_{M_i}(x) \leq 2^{|i|}]\} \\ CM_{RE} &\stackrel{\text{def}}{=} \{i \mid (\exists c)(\forall x) [T_{M_i}(x) \leq c]\} \end{aligned}$$

Now we define the creativeness for the classes EXP, NEXP and r.e. based on these presentations of CONST — our creativeness for EXP is defined to be (p, CM_E) -creativeness; for NEXP it is defined to be (p, CM_{NE}) -creativeness and for r.e. it is (rec, CM_{RE}) -creativeness.

Theorem 3.16 (i) A is (rec, N) -creative $\Leftrightarrow A$ is (rec, CM_{RE}) -creative.

(ii) A is (p, NPM) -creative $\Leftrightarrow A$ is (p, CM_{NE}) -creative.

(iii) A is (p, PM) -creative $\Leftrightarrow A$ is (p, CM_E) -creative.

Proof Sketch. (As we had observed in section 2.2, the left hand side of each of the assertions above deals with standard creative sets as denoted in our notation. Therefore, the theorem says that the standard notion of creativeness coincides with our notion for the classes EXP, NEXP and r.e.) We prove part (ii) of the theorem. The other two parts will follow similarly.

(ii) The forward implication is obvious. To prove the reverse implication we use a form of recursion theorem for bounded complexity classes. Let s_0, s_1, \dots be the sequence of all 2-ary polynomial time functions. Let h be the (p, CM_{NE}) -productive function for the set A . Define TM $M_{g(j,i)}$ as — On input x , compute $h(s_j(j, i))$; reject if $x \neq h(s_j(j, i))$. Otherwise, accept iff TM M_i accepts x in $p_i(|x|)$ steps.

Choose g such that $(|j| + |i|)^2 \leq |g(j, i)|$ and g is computable in polynomial time. Therefore, g itself will be a 2-ary polynomial time function. Let $g = s_{j_0}$ and q

$\lambda i. g(j_0, i) = \lambda i. s_{j_0}(j_0, i)$. So, for all $i \in NPM$, $h(q(i)) \in W_i$ iff $h(q(i)) \in W_{q(i)}$ and $T_{M_{q(i)}}(x) \leq O[|q(i)|^{c_1} \cdot |i|] \leq 2^{c_2 \cdot |i| \cdot \log |i|} \leq 2^{|q(i)|}$ (a.e. i). Therefore, $q(i) \in CM_{NE}$ (a.e. i) and function $h \circ q$ is the required (p, NPM) -productive function almost everywhere. As before, this can be easily modified to yield a total productive function. \square

Most of the theorems that hold for the class NP translate to these classes as well. In particular, one can easily show, by a simple translation of the proof of Theorem 2.4, that (p, CM_E) -creative and (p, CM_{NE}) -creative sets are EXP- and NEXP-hard respectively.

Theorem 3.17 (i) *If A is (p, CM_{NE}) -creative then A is NEXP-hard.*

(ii) *If A is (p, CM_E) -creative then A is EXP-hard.*

Proof Sketch. The proof is identical to the proof of the Theorem 2.4 except for time bounds on the TMs involved. For NEXP-case, the NDTM M_B computing language B will take $2^{p_1(|x|)}$ steps on input x and therefore $T_{M_{q(x)}}(z) \leq 2^{p_2(|x|)}$. Defining function q as before, we get $T_{M_{q(x)}}(z) \leq 2^{|q(x)|}$ and $W_{q(x)} = W_{g(x)}$. So, $q(x) \in CM_{NE}$ and using the productiveness property we get $x \in B$ iff $h(q(x)) \in A$, where h is the productive function for A . For EXP-case, all the TMs involved will become deterministic, the rest of the proof remaining unchanged. \square

Corollary 3.18 (i) *If A is (p, NPM) -creative then A is NEXP-hard.*

(ii) *If A is (p, PM) -creative then A is EXP-hard.*

Proof. Follows from the above two Theorems. \square

Wang [14] had posed the statement of the above Corollary as an open question. He had shown that all EXP- and NEXP-hard sets are respectively (p, PM) - and (p, NPM) -creative. Therefore, creativeness and hardness coincide for the classes EXP and NEXP.

The Corollary also shows that no (p, NPM) -creative set can be in NP. As far as p-isomorphism of these complete degrees is concerned, we could not obtain any new result. The existing results for these classes show that all complete sets for EXP are complete under one-one, size-increasing reductions [2] while all complete sets for NEXP are complete under one-one, exponentially honest reductions [5].

3.4 Creative sets in PSPACE

In this section, we define and study creative sets for the class PSPACE. As in the previous section, define

$$CM_{PSPACE} \stackrel{\text{def}}{=} \{i \mid M_i \text{ a DTM and } (\forall x)[S_{M_i}(x) \leq |i|]\}$$

We write (lg, CM_{PSPACE}) -creative sets for creative sets over CM_{PSPACE} with logspace computable productive function. This is the definition we will use to study logspace-complete sets for PSPACE. As for EXP and NEXP, we show that (lg, CM_{PSPACE}) -sets are *exactly* those sets that are logspace-hard for PSPACE. We also show that all logspace-complete sets for PSPACE are complete under one-one and size-increasing reductions.

Define set

$$K_{PSPACE} \stackrel{\text{def}}{=} \{i \mid M_i \text{ a DTM and accepts } i \text{ in } |i| \text{ space}\}$$

K_{PSPACE} is the canonical (lg, CM_{PSPACE}) -creative set.

Proposition 3.19 *K_{PSPACE} is in PSPACE and is (lg, CM_{PSPACE}) -creative with identity productive function.*

The following theorem shows that all (lg, CM_{PSPACE}) -creative sets in PSPACE are complete for PSPACE under size-increasing logspace reductions.

Theorem 3.20 (i) If A is (lg, CM_{PSPACE}) -creative then A is PSPACE-hard under logspace reductions.

(ii) If A is (lg, CM_{PSPACE}) -creative and $A \in PSPACE$ then A is PSPACE-complete under size-increasing logspace reductions.

Proof. The proof moves along the same lines as the previous two hardness proofs. Part (i) can be proved by simply making the necessary substitutions in the proof of Theorem 2.4. The only point to note is that all constructed index manipulation functions, e.g. g and q , are logspace functions. We only give the proof for part (ii) which utilizes the creativeness property to get a size-increasing reduction. This proof is similar to the proof of Theorem 3.1, the difference being that since PSPACE is a deterministic class, we get size-increasing reductions instead of exponentially honest.

Let $B \in PSPACE$ and recognized by DTM M_B working in space bounded by the polynomial p_r . Since $A \in PSPACE$ there will be a DTM M_A that recognizes A in space bounded by some polynomial, say p_s . For every x , define TM $M_{g(x)}$ as —

On input z , if $|z| \leq |x|$, run TM M_A on z and accept iff it rejects z . If $|z| > |x|$ then run TM M_B on x and accepts iff it accepts x .

$S_{M_{g(x)}}(z) \leq O[p_s(|x|) + p_r(|x|)] \leq p_k(|x|)$ for some polynomial p_k . Letting $q = \lambda x. g(x)0^{p_k(|x|)}$ as before, we get $S_{M_{q(x)}}(z) \leq |q(x)|$. Thus, for every $x, q(x) \in CM_{PSPACE}$. By construction we have,

$$W_{q(x)} = \begin{cases} \{\bar{A}\}_{\leq |x|} & \text{if } x \notin B \\ \{\bar{A}\}_{\leq |x|} \cup \Sigma_{>|x|} & \text{if } x \in B \end{cases}$$

Let h be the productive function for A and $f \stackrel{\text{def}}{=} h \circ q$. We note that f is a logspace computable function. Suppose for some x , $|f(x)| \leq |x|$. Then, we get $f(x) \in W_{q(x)}$ iff $f(x) \in \bar{A}$, by the construction of q and $f(x) \in W_{q(x)}$ iff $f(x) \in A$, by the creativeness

of A , a contradiction. So, f is size-increasing and then it is easy to see that, $x \in B$ iff $f(x) \in A$. \square

The technique used to prove the following theorem is similar to the one discussed at the beginning of the section 2.4.

Theorem 3.21 *If A is logspace-hard for PSPACE then A is (lg, CM_{PSPACE}) -creative.*

Proof. Let the logspace computable function f reduce K_{PSPACE} to A . Define DTM $M_{g(i)}$ as —

On input z , reject if $|z| > |i|^2$; else compute $f(z)$ and accept iff DTM M_i accepts $f(z)$.

If $M_{g(i)}$ computes whole of $f(z)$ before starting the computation of M_i then, since $f(z)$ needs to be stored, the space required may become a polynomial in $|z|$. This will clearly not be acceptable if we want to show $g(i) \in CM_{PSPACE}$ for every $i \in CM_{PSPACE}$. So, $M_{g(i)}$ does not compute $f(z)$ beforehand, instead it starts the computation of M_i and computes bits of $f(z)$ *on demand*, i.e., whenever M_i requires a bit of $f(z)$, it computes $f(z)$ till the required bit is obtained *without* storing the intermediate bits. This way it avoids storing $f(z)$. For such a $M_{g(i)}$ and suitably padded g , it is easy to see that if $i \in CM_{PSPACE}$ then $S_{M_{g(i)}}(z) \leq O[|i|] \leq |g(i)| = |i|^2$ a.e. Thus, for almost all $i \in CM_{PSPACE}$, $g(i) \in CM_{PSPACE}$ and g is a logspace computable function.

Now, for almost all $i \in CM_{PSPACE}$, $f(g(i)) \in A$ iff $g(i) \in K_{PSPACE}$ (by reduction) iff $g(i) \in W_{g(i)}$ iff $f(g(i)) \in W_i$. Therefore, $f \circ g$ is a productive function for A almost everywhere and can be easily modified to yield the required productive function. \square

Corollary 3.22 *A is (lg, CM_{PSPACE}) -creative iff A is logspace-hard for PSPACE.*

Corollary 3.23 *Logspace-complete sets for PSPACE are (1) complete under size-increasing logspace reductions and (2) (lg, CM_{PSPACE}) -creative with size-increasing productive functions.*

Proof. Follows from the above two theorems and the fact that if there is a size-increasing logspace reduction from K_{PSPACE} to a language, the language will have a size-increasing productive function. \square

By a simple use of recursion theorem for bounded complexity classes, one can show that all logspace-complete sets for PSPACE are one-one logspace-complete as well.

Theorem 3.24 *Logspace-complete sets for PSPACE are complete under one-one and size-increasing logspace reductions.*

Proof. Let set A be logspace-complete for PSPACE. We will show that K_{PSPACE} is reducible to A by a one-one, size-increasing logspace reduction. Let h be the size-increasing productive function for A .

Let r_0, r_1, \dots be an enumeration of all 2-ary logspace computable functions. Define the DTM $M_{g(k,i)}$ as —

On input x , reject if $|x| \leq |i|$. Otherwise, check if there is some j , $j < i$ such that $h(r_k(k, j)) = h(r_k(k, i))$. If yes then accept iff DTM M_i does not accept i within $|i|$ space. If no then accept iff DTM M_i accepts i within $|i|$ space.

Function g itself is a 2-ary, logspace computable function. Let $g = r_e$ for some e and $q = \lambda i. g(e, i) = \lambda i. r_e(e, i)$. By computing function $h \circ r_e$ on demand, as done in the previous proof, one can ensure that $S_{M_{g(i)}}(x) \leq O(|i|) \leq |q(i)|$ a.e. for suitably padded q . So, for every $i > i_0$ (for some fixed i_0), $q(i) \in CM_{PSPACE}$. Let function $f \stackrel{\text{def}}{=} h \circ q$.

Claim : f is a one-one, size-increasing logspace computable function almost everywhere.

Proof. That f is logspace computable and size-increasing follows directly from the fact that both h and q are logspace computable and size-increasing. Suppose f is not one-one on infinitely many inputs. Let i_0 and $j_0, i_0 > j_0 > i_s$, be the least two numbers for which $f(i_0) = f(j_0)$. Since f is size-increasing, DTM $M_{q(j_0)}$ will accept $f(j_0)$ iff M_{j_0} accepts $f(j_0)$ and DTM $M_{q(i_0)}$ will accept $f(i_0)$ iff M_{i_0} rejects $f(i_0)$. In other words, $M_{q(j_0)}$ accepts $f(j_0)$ iff $M_{q(i_0)}$ rejects $f(i_0)$. Using creativeness of A , we have that $f(i) \in W_{q(i)}$ iff $f(i) \in A$ for every $i > i_s$. Since, $f(i_0) = f(j_0) = x_0$ (say), we get $x_0 \in A$ iff $x_0 \in W_{q(i_0)}$ iff $x_0 \notin W_{q(j_0)}$ contradicting the creativeness of A . Therefore, f must be one-one almost everywhere. \square

Function f , being one-one and size-increasing almost everywhere, will clearly be a reduction of K_{PSPACE} to A except for finitely many inputs. This can be easily modified to yield a one-one and size-increasing reduction. And since K_{PSPACE} is complete for PSPACE under one-one, size-increasing logspace reductions, so will be A . \square

Chapter summary

In this chapter, we have shown that 1-L-complete sets for NP are p-isomorphic. This result can also be generalized to many other classes. We also have obtained new definitions of creativeness for the classes PSPACE, EXP, NEXP and r.e. For EXP, NEXP and r.e. these definitions turn out to be equivalent to the existing ones in literature. We show that many-one complete sets for EXP and NEXP classes are characterized by their respective creativeness properties. Similarly, logspace-complete sets for PSPACE are characterized by the creativeness property for PSPACE. We also have shown, using creativeness property, that logspace-complete sets for the class PSPACE are complete under one-one and size-increasing logspace reductions.

Chapter 4

The Universal Relation

4.1 Introduction

In this chapter, we define certain kind of relations witnessing NP-complete languages. The motivation for the definition comes from the fact that reductions amongst natural NP-complete problems are usually very structured and well behaved. These reductions, in a way, code the entire 'structure' of the input instance in the output instance. We have tried to capture this property in the following way. Consider the reduction of a natural complete language A to another natural language B . Given an instance of A , the process of constructing the corresponding instance of B can be divided in two stages. In the first stage a number of copies of a particular element of A are taken and successively joined together so that the number of solutions keep multiplying. This process is continued till an instance is obtained with sufficiently large number of solutions. The purpose of this stage is simply to get an instance in which the structure of the input instance can be coded. The actual coding of the input instance is carried out in the second stage. Here, the process is that of successively *disallowing* certain solutions from the instance obtained at the end of the first stage based on the structure

of the input instance. At the end of the second stage, one gets an instance that has a solution if and only if the input instance has a solution and further there is a nice correspondence between solutions of the two instances.

We define two operators capturing the processes of the above two stages. As the set of solutions for an instance of a language depends on the relation witnessing the language, these two operators are defined for a relation instead of a language. The first operator, called *join*, increases the number of solutions and corresponds to the first stage of the above process. The second operator, called *equivalence*, decreases the number of solutions and corresponds to the second stage.

The chapter is organized as follows. Section 4.2 deals with notations. In section 4.3, we give the definitions of the two operators and the universal relation. In section 4.4, we prove that universal relations witness NP-complete sets and also give a characterization of these relations. Section 4.5 contains some results that are useful in proving a given relation universal. Section 4.6 is devoted to providing examples of relations that are universal while the section 4.7 gives examples of relations that are not universal. In this section we also define generalized universal relations, the class of which strictly contains the class of universal relations. The two operators that we define are closely related to paddability and d-self-reducibility respectively. In section 4.8, we investigate this relationship. Section 4.9 proves some further results concerning universal and generalized universal relations.

4.2 Notations

All the strings in this chapter are assumed to be over the alphabet $\Sigma = \{0, 1\}$. If x is a string over Σ of length n then we use x^i to denote the i^{th} bit of x , i.e., $x = x^1x^2 \dots x^n$ with each $x^i \in \Sigma$. We shall use $\alpha, \beta, \gamma, \dots$ etc. to denote finite sequences of *distinct*

positive integers, and if α is the sequence n_1, \dots, n_k , i.e., $\alpha = \{n_i\}_1^k$ then α^j will denote n_j , the j^{th} element of the sequence. For sequence α , $|\alpha|$ denotes the number of elements in α . For any positive integer m and a sequence α , $\alpha + m \stackrel{\text{def}}{=} \{m + \alpha^i\}_1^{|\alpha|}$, that is, the sequence obtained from α by adding m to each element of α . α, β denotes the sequence obtained by *concatenating* β to the right of α , further, concatenation of α and β will be defined only if they have no element in common. For sequences α and β , if the largest element of β is less than or equal to $|\alpha|$, then the sequence $\alpha | \beta$ is defined as follows : $\alpha | \beta \stackrel{\text{def}}{=} \{m_i\}_1^{|\beta|}$ where $m_i = \alpha^{\beta^i}$.

Let S be a set of equal length strings. We define the following *masking* operation on such sets.

The Mask Operation [$Mask(S, \alpha)$] : Let $S \subseteq \Sigma_m$ and $\alpha = \{n_i\}_1^k$ with $1 \leq n_i \leq m$ for $1 \leq i \leq k$. Then $Mask(S, \alpha) \stackrel{\text{def}}{=} \{y \mid (\exists x) x \in S \wedge y = x^{n_1} x^{n_2} \dots x^{n_k}\}$. We refer to $Mask(S, \alpha)$ as the set S *restricted to* α . The following identity is obvious.

IDENTITY : $Mask(S, \alpha | \beta) = Mask(Mask(S, \alpha), \beta)$

For every set A in NP, by definition, there exists a polynomial time verifiable binary relation R , $R \subseteq \Sigma^* \times \Sigma^*$, and a polynomial p such that

$$x \in A \Leftrightarrow (\exists s)[|s| \leq p(|x|) \wedge xRs]$$

We will refer to the strings s such that xRs and $|s| \leq p(|x|)$, as *solutions* of x . The relation R is said to be a relation *witnessing* A to be in NP, or simply, *witnessing* A . Define the *solution set* of x , $sol_R(x) \stackrel{\text{def}}{=} \{s \mid xRs \wedge |s| \leq p(|x|)\}$. Without loss of generality, we can assume that all the solutions of any string are of the same length and that this length is computable in polynomial time. We call such a relation *admissible* and if R is an admissible relation then there is a polynomial time computable function $sol-len_R$ such that $(\forall x)[sol-len_R(x) \geq 1 \ \& \ [sol_R(x) \neq \emptyset \Rightarrow sol_R(x) \subseteq \Sigma_{= sol-len_R(x)}]]$.

4.3 Universal relations

To begin with, we define below two operators *join* and *equivalence* for an admissible relation R . Each operator has two functions associated with it, one called the *instance function* and other the *sequence function*. The two instance functions take one and two instances respectively and produce a new instance such that the solution set of the new instance is related to the solution set of the input instance(s) in a certain specified way. This relation can be captured by using the corresponding sequence functions.

Definition 4.1 Let R be an admissible relation. R has a *join operator* if there exist polynomial time computable functions $join_R$ and $join-seq_R$ such that

1. $join_R(x, y) = z$ and $join-seq_R(x, y) = \alpha$ where $x, y, z \in \Sigma^*$ and $|\alpha| = sol-len_R(x) + sol-len_R(y)$.
2. $Mask(sol_R(z), \alpha) = \{st \mid s \in sol_R(x) \wedge t \in sol_R(y)\}$.

Definition 4.2 Let R be an admissible relation. R has an *equivalence operator* if there exist polynomial time computable functions equ_R and $equ-seq_R$ such that

1. $equ_R(x, i, j) = z$ and $equ-seq_R(x, i, j) = \alpha$ where $x \in \Sigma^*$, $1 \leq i \neq j \leq sol-len_R(x)$ and $|\alpha| = sol-len_R(x)$.
2. $Mask(sol_R(z), \alpha) = \{s \mid s \in sol_R(x) \wedge s^i = s^j\}$. (We note that in every solution of z the two bits α^i and α^j will have the same value.)

We now define an instance with a certain solution set. This instance, as we shall see, will be the starting point in the construction of the reduction of a known NP-complete problem to the set.

Definition 4.3 Let R be an admissible relation witnessing the language A . R has a *building block* if there exists an element in A called $block_R$ and four positive integers bit_1 , bit_2 , bit_3 and bit_4 such that

$$Mask(sol_R(block_R), \alpha) = \{1001, 1010, 1011, 0100, 0101, 0110, 0111\}$$

where $\alpha = bit_1, bit_2, bit_3, bit_4$. In other words, the solution set of $block_R$ has at least one solution for each assignment of the three bit positions bit_2 , bit_3 and bit_4 except the assignment 000 and in each of these solutions, the symbol at the bit position bit_1 is the complement of the symbol at the bit position bit_2 .

We illustrate the above definitions by taking as example the standard relation for Satisfiability problem (SAT). Let the relation R_{SAT} be defined as : $xR_{SAT}s$ iff $|s| = n$ (n is the number of variables in x) and s codes a satisfying assignment of x with $s^i = 1$ iff variable v_i is true in the assignment.

R_{SAT} has a join operator — let $join_{R_{SAT}}(x, y) \stackrel{\text{def}}{=} z$ where z is obtained by taking the conjunction of x and y' , and y' is obtained from y by renaming all the variables to make them different from those of x ; $join-seq_{R_{SAT}}(x, y) \stackrel{\text{def}}{=} \{i\}_1^n$, where $n = sol-len_{R_{SAT}}(x) + sol-len_{R_{SAT}}(y)$. Clearly, both the functions are computable in polynomial time and the solutions of $join_{R_{SAT}}(x, y)$ will be a concatenation of the solutions of x and y .

R_{SAT} has an equivalence operator as well — let $equ_{R_{SAT}}(x, a, b) \stackrel{\text{def}}{=} x'$ where x' is obtained by making the variables v_a and v_b equivalent. In other words, conjunct x with the clauses $(\bar{v}_a \vee v_b)$ and $(v_a \vee \bar{v}_b)$; $equ-seq_{R_{SAT}}(x, a, b) \stackrel{\text{def}}{=} \{i\}_1^n$, where $n = sol-len_{R_{SAT}}(x)$. Both the functions can be computed in polynomial time and the required correspondence between the solutions of x and x' is clearly seen to hold.

R_{SAT} has the following building block — $block_{R_{SAT}} : (v_1 \vee v_2) \wedge (\bar{v}_1 \vee \bar{v}_2) \wedge (v_2 \vee v_3 \vee v_4)$ with $bit_1 = 1$, $bit_2 = 2$, $bit_3 = 3$ and $bit_4 = 4$.

We will need to apply the functions $join_R$ and equ_R iteratively. We will also require that the length of the resulting instance is bounded by a polynomial in the length of the

original instance(s) and the number of applications. This condition is not guaranteed by the above definitions as the length of the resulting instance may become exponential in the number of applications. To get around this problem, we define iterative versions of these two operators satisfying the constraints.

Definition 4.4 Let R be an admissible relation.

1. R has an *iterative join operator* if there exist polynomial time computable functions $iter-join_R$ and $iter-join-seq_R$ such that

$$(i) \quad iter-join_R(\langle x_1, \dots, x_n \rangle, n) = z \text{ and } iter-join-seq_R(\langle x_1, \dots, x_n \rangle, n) = \alpha$$

where $x_1, \dots, x_n, z \in \Sigma^*$ and $|\alpha| = \sum_{k=1}^n sol-len_R(x_k)$.

$$(ii) \quad Mask(sol_R(z), \alpha) = \{s_1 s_2 \dots s_n \mid (\forall i \leq n) s_i \in sol_R(x_i)\}.$$

2. R has an *iterative equivalence operator* if there exist polynomial time computable functions $iter-equ_R$ and $iter-equ-seq_R$ such that

$$(i) \quad iter-equ_R(x, n, \langle i_1, \dots, i_n \rangle, \langle j_1, \dots, j_n \rangle) = z \text{ and}$$

$$iter-equ-seq_R(x, n, \langle i_1, \dots, i_n \rangle, \langle j_1, \dots, j_n \rangle) = \alpha$$

where $x \in \Sigma^*$, $1 \leq i_1, \dots, i_n, j_1, \dots, j_n \leq sol-len_R(x)$, $i_m \neq j_m$ for each $1 \leq m \leq n$ and $|\alpha| = sol-len_R(x)$.

$$(ii) \quad Mask(sol_R(z), \alpha) = \{s \mid s \in sol_R(x) \wedge (\forall m \leq n) s^{i_m} = s^{j_m}\}.$$

(As the pairing function used is an onto function, the argument n in the above definitions helps in inverting correctly.)

When the functions $join_R$ and equ_R do not grow too fast then the corresponding iterative operators can be constructed by an iterative applications of these functions.

Lemma 4.5 Let R be an admissible relation.

- (i) If R has a join operator satisfying $(\forall x)(\forall y)[|join_R(x, y)| \leq c.(|x| + |y|)]$ for some constant c , then R has an iterative join operator.
- (ii) If R has an equivalence operator satisfying $(\forall x)(\forall i)(\forall j)[|equ_R(x, i, j)| \leq p(p^{-1}(|x|) + c)]$ for some polynomial p and constant c , then R has an iterative equivalence operator.

Proof. We give a recursive definition of the iterative versions of the two operators.

- (i) Letting $m = \lceil n/2 \rceil$ and denoting

$$\begin{aligned}
 y_1 &= iter-join_R(\langle x_1, \dots, x_m \rangle, m) \\
 \alpha &= iter-join-seq_R(\langle x_1, \dots, x_m \rangle, m) \\
 y_2 &= iter-join_R(\langle x_{m+1}, \dots, x_n \rangle, n - m) \\
 \beta &= iter-join-seq_R(\langle x_{m+1}, \dots, x_n \rangle, n - m) \\
 length &= \sum_{k=1}^m sol-len_R(x_k) \\
 \gamma &= \alpha, length + \beta
 \end{aligned}$$

define,

$$\begin{aligned}
 iter-join_R(\langle x_1, \dots, x_n \rangle, n) &\stackrel{\text{def}}{=} \begin{cases} x_1 & \text{if } n = 1 \\ join_R(x_1, x_2) & \text{if } n = 2 \\ join_R(y_1, y_2) & \text{if } n > 2 \end{cases} \\
 iter-join-seq_R(\langle x_1, \dots, x_n \rangle, n) &\stackrel{\text{def}}{=} \begin{cases} \{t\}_i^{sol-len_R(x_1)} & \text{if } n = 1 \\ join-seq_R(x_1, x_2) & \text{if } n = 2 \\ join-seq_R(y_1, y_2) \mid \gamma & \text{if } n > 2 \end{cases}
 \end{aligned}$$

Both the functions will be computable in polynomial time since, by induction, it is easy to show that $|y_1| \leq c.m^{log c}.(|x_1| + \dots + |x_m|)$ and $|y_2| \leq c.(n-m)^{log c}.(|x_{m+1}| + \dots + |x_n|)$.

(ii) Denoting

$$\begin{aligned}
 y &= iter-equ_R(x, n-1, \langle i_1, \dots, i_{n-1} \rangle, \langle j_1, \dots, j_{n-1} \rangle) \\
 \alpha &= iter-equ-seq_R(x, n-1, \langle i_1, \dots, i_{n-1} \rangle, \langle j_1, \dots, j_{n-1} \rangle) \\
 i &= \alpha^{i_n} \\
 j &= \alpha^{j_n}
 \end{aligned}$$

define,

$$\begin{aligned}
 iter-equ_R(x, n, \langle i_1, \dots, i_n \rangle, \langle j_1, \dots, j_n \rangle) &\stackrel{\text{def}}{=} equ_R(y, i, j) \\
 iter-equ-seq_R(x, n, \langle i_1, \dots, i_n \rangle, \langle j_1, \dots, j_n \rangle) &\stackrel{\text{def}}{=} \begin{cases} equ-seq_R(x, i, j) & \text{if } n = 1 \\ equ-seq_R(y, i, j) \mid \alpha & \text{if } n > 1 \end{cases}
 \end{aligned}$$

It is easily verifiable that $|y| \leq p(|x| + (n-1)*c)$ and therefore, both the functions will be computable in polynomial time.

□

Almost all natural problems having join and equivalence operators satisfy these constraints and therefore will have the iterative versions of these operators as well. For example, the functions $join_{R_{SAT}}$ and $equ_{R_{SAT}}$ trivially satisfy these constraints.

Now we define our universal relation.

Definition 4.6 Let R be an admissible relation. R is a *universal* relation if it has a building block and iterative join and iterative equivalence operators.

Relation R_{SAT} is obviously universal because of the above discussion.

4.4 Universal relations witness NP-complete sets

We are ready now to prove basic results.

Theorem 4.7 *If R is a universal relation witnessing the set A then A is NP-complete.*

Proof. We will reduce the NP-complete set 3SAT to A . (3SAT is a subset of SAT and contains all those satisfiable boolean formulas that have exactly three literals per clause.)

Let x be an instance of 3SAT with n variables and m clauses. Define

$$block_R^k \stackrel{\text{def}}{=} \begin{cases} \langle block_R, block_R^{k-1} \rangle & \text{if } k > 1 \\ block_R & \text{otherwise} \end{cases}$$

and,

$$\begin{aligned} y &\stackrel{\text{def}}{=} \text{iter-join}_R(block_R^{n+m}, n+m) \\ \alpha &\stackrel{\text{def}}{=} \text{iter-join-seq}_R(block_R^{n+m}, n+m) \end{aligned}$$

By definition, we have, $\text{Mask}(\text{sol}_R(y), \alpha) = \{s_1 \cdots s_{n+m} \mid (\forall i \leq n+m) s_i \in \text{sol}_R(block_R)\}$.

We will choose certain bit positions of solutions of y from the sequence α and apply the iterative equivalence operator on them.

Let $\text{length} = \text{sol-len}_R(block_R)$ and for each i and j , $1 \leq i \leq n$, $1 \leq j \leq m$, define

$$\begin{aligned} \text{var}_i &= \alpha^{(i-1)*\text{length}+\text{bit}_1} \\ \text{cvar}_i &= \alpha^{(i-1)*\text{length}+\text{bit}_2} \\ \text{cl}_j^1 &= \alpha^{(n+j-1)*\text{length}+\text{bit}_2} \\ \text{cl}_j^2 &= \alpha^{(n+j-1)*\text{length}+\text{bit}_3} \\ \text{cl}_j^3 &= \alpha^{(n+j-1)*\text{length}+\text{bit}_4} \end{aligned}$$

For these positions, it is easy to see that letting

$$\beta = \{var_i\}_1^n, \{cvar_i\}_1^n, \{cl_j^1, cl_j^2, cl_j^3\}_1^m$$

we get,

$$Mask(sol_R(y), \beta) = \{s\bar{s}t_1t_2 \cdots t_m \mid s \in \Sigma_{=n} \wedge (\forall i \leq m)t_i \in \Sigma_{=3} - \{000\}\}$$

where \bar{s} stands for the bitwise complement of the string s . Define

$$z \stackrel{\text{def}}{=} iter-equ_R(y, 3m, \langle i_1, \dots, i_{3m} \rangle, \langle cl_1^1, cl_1^2, cl_1^3, \dots, cl_m^1, cl_m^2, cl_m^3 \rangle)$$

where for $1 \leq k \leq 3$ and $1 \leq j \leq m$, $i_{3(j-1)+k} = var_l$ if in the k^{th} position of the j^{th} clause of x the l^{th} variable occurs *positively*, $cvar_l$ if it occurs *negatively*. Thus, if the j^{th} clause of x is $v_{l_1} \vee \bar{v}_{l_2} \vee v_{l_3}$, then equating of bit positions $i_{3(j-1)+1}$, $i_{3(j-1)+2}$, $i_{3(j-1)+3}$ with cl_j^1 , cl_j^2 , cl_j^3 will disallow all those solutions that have zeroes in bit positions var_{l_1} , $cvar_{l_2}$ and var_{l_3} . So, the effect of this on the solution set restricted to the sequence var_1, \dots, var_n will be exactly the same as the effect of the j^{th} clause on the solution set of x . Therefore, letting

$$\gamma = iter-equ-seq_R(y, 3m, \langle i_1, \dots, i_{3m} \rangle, \langle cl_1^1, \dots, cl_m^3 \rangle)$$

and $\delta = \{\gamma^{\beta^i}\}_1^n$, we get $Mask(sol_R(z), \delta) = sol_{R_{3SAT}}(x)$ which gives $x \in 3SAT \Leftrightarrow z \in A$.

We can see from the above discussion, and from the definitions of the two operators, that z can be computed in time bounded by a polynomial in $|x|$. Therefore $f \stackrel{\text{def}}{=} \lambda x. z$ is a polynomial time reduction of 3SAT to A . \square

The universal relations derive their name from the following property.

Theorem 4.8 *R is a universal relation iff for every admissible relation Q , there are polynomial time functions f_R^Q and $f_seq_R^Q$ such that $(\forall x)[Mask(sol_R(f_R^Q(x)), \alpha) = sol_Q(x)]$, where $\alpha = f_seq_R^Q(x)$.*

Proof. (\Rightarrow) An examination of Cook's encoding of computations of NDTM in instances of SAT [4] reveals the following : there exists polynomial time computable functions f_{SAT}^M and f_{seqSAT}^M such that the set $sol_{R_{SAT}}(f_{SAT}^M(x))$ when restricted to the sequence $f_{seqSAT}^M(x)$, has exactly those strings that are accepting computations of TM M on x . Given an admissible relation Q witnessing some set A , one can design an NDTM that accepts x by first guessing a solution of x with respect to Q and then verifying it. Moreover, the guess of the solution will be in some *fixed* positions of the accepting computation of the NDTM on x . Thus, one can construct polynomial time functions $f_{R_{SAT}}^Q$ and $f_{seqR_{SAT}}^Q$ such that these satisfy the property given in the theorem statement. From this, such functions for relation R_{3SAT} can be easily obtained. In the proof of the previous theorem, it was shown that if R is a universal relation then it has the above two functions when $Q = R_{3SAT}$. A composition of these functions with the functions for R_{3SAT} will yield the functions required in the statement of the theorem.

(\Leftarrow) We will construct the two operators for the relation R using the two operators for R_{SAT} and functions $f_{R_{SAT}}^R$, $f_{seqR_{SAT}}^R$, $f_R^{R_{SAT}}$ and $f_{seqR}^{R_{SAT}}$. Let $y_i = f_{R_{SAT}}^R(x_i)$ and $\alpha_i = f_{seqR_{SAT}}^R(x_i)$ for $1 \leq i \leq n$. We have, $Mask(sol_{R_{SAT}}(y_i), \alpha_i) = sol_R(x_i)$ for $1 \leq i \leq n$. We will use the iterative join operator of R_{SAT} on these y_i and the use $f_R^{R_{SAT}}$ on the output to get the iterative join operator for R . Let $Sum(i) = \sum_{j=1}^i sol_len_{R_{SAT}}(y_j)$ and define sequence $\beta = \alpha_1, (Sum(1) + \alpha_2), \dots, (Sum(n-1) + \alpha_n)$. Then, letting $y = iter_join_{R_{SAT}}(\langle y_1, \dots, y_n \rangle, n)$ and $\gamma = iter_join_seq_{R_{SAT}}(\langle y_1, \dots, y_n \rangle, n)$, we get $Mask(sol_{R_{SAT}}(y), \gamma \mid \beta) = \{s_1 \cdots s_n \mid (\forall i \leq n) s_i \in sol_R(x_i)\}$. Therefore,

$$\begin{aligned} iter_join_R(\langle x_1, \dots, x_n \rangle, n) &\stackrel{\text{def}}{=} f_R^{R_{SAT}}(y) \\ iter_join_seq_R(\langle x_1, \dots, x_n \rangle, n) &\stackrel{\text{def}}{=} f_{seqR}^{R_{SAT}}(y) \mid (\gamma \mid \beta) \end{aligned}$$

will give the iterative join operator for R .

One can similarly construct iterative equivalence operator and the building block for the relation is given by $block_R \stackrel{\text{def}}{=} f_R^{R_{SAT}}(block_{R_{SAT}})$. \square

Corollary 4.9 *Let R be a universal relation and W be a set of equal length strings. Then there is an instance x and a sequence α such that $\text{Mask}(\text{sol}_R(x), \alpha) = W$.*

Proof Sketch. One can easily construct a SAT instance y such that $\text{sol}_{R_{\text{SAT}}}(y) = W$. Now, using the above Theorem we get an instance x and a sequence α satisfying the required property. \square

4.5 Some results useful in application

We shall find the following useful in showing specific relations to be universal.

Definition 4.10 Let R be an admissible relation. R has a *negation operator* if there exist polynomial time computable functions neg_R and neg-seq_R such that

1. $\text{neg}_R(x, i, j) = z$ and $\text{neg-seq}_R(x, i, j) = \alpha$ where $x \in \Sigma^*$, $1 \leq i, j \leq \text{sol-len}_R(x)$, $i \neq j$ and $|\alpha| = \text{sol-len}_R(x)$.
2. $\text{Mask}(\text{sol}_R(z), \alpha) = \{s \mid s \in \text{sol}_R(x) \wedge s^i \neq s^j\}$.

The iterative negation operator consisting of functions iter-neg_R and iter-neg-seq_R is defined in a similar way. One can show that relations having a building block, iterative join operator and iterative negation operator instead of iterative equivalence operator will still be universal.

Theorem 4.11 *Let R be an admissible relation. R is universal iff it has a building block, iterative join and negation operators.*

Proof. (\Rightarrow) The idea of the proof is as follows. Recall that a block_R has two complementary bits, namely bit_1 and bit_2 . To get $\text{neg}_R(x, i, j)$, we can join x with block_R

and then use equ_R to equate the complementary bits with i and j respectively. The iterative version can be similarly obtained.

Define $iter-neg_R(x, n, \langle i_1, \dots, i_n \rangle, \langle j_1, \dots, j_n \rangle) \stackrel{\text{def}}{=} z$, where z is constructed in the following way — first compute

$$y_1 = iter-join_R(block_R^n, n) [block_R^n \text{ as defined in the proof of Theorem 4.7}]$$

$$y_2 = iter-join_R(\langle x, y_1 \rangle, 2)$$

$$\alpha = iter-join-seq_R(block_R^n, n)$$

$$\beta = iter-join-seq_R(\langle x, y_1 \rangle, 2)$$

Let $i'_k = \beta^{i_k}$, $j'_k = \beta^{j_k}$, $var_k = (\beta \mid \alpha)^{(k-1)*length+bit_1}$ and $cvar_k = (\beta \mid \alpha)^{(k-1)*length+bit_2}$ for $1 \leq k \leq n$, where $length = sol-len_R(block_R)$. Compute

$$z = iter-equ_R(y, 2n, \langle i'_1, \dots, i'_n, j'_1, \dots, j'_n \rangle, \langle var_1, \dots, var_n, cvar_1, \dots, cvar_n \rangle)$$

It is easy to see that z has the desired properties. Function $iter-neg-seq_R$ can be easily obtained from the above construction.

(\Leftarrow) To construct the iterative equivalence operator from iterative negation and join operators a similar idea is used. Bits i and j of instance x are equated by first joining x with $block_R$ and then negating bits i and j with bit_1 of $block_R$. \square

For a number of NP-complete problems, the operators can be more easily defined over an infinite subset S of Σ^* . We can extend them to be over Σ^* using the following theorem.

Definition 4.12 Let $S \subseteq \Sigma^*$ and R be an admissible relation. R is S -universal if

1. $block_R \in S$.
2. $iter-join_R(\langle x_1, \dots, x_n \rangle, n) = z$ with $x_1, \dots, x_n, z \in S$.

have a universal relation. The aim of this section is to show that even the *most obvious* relations of many natural sets are universal.

In the examples below, we give the $join_R$ and neg_R functions, the corresponding sequence functions will be obvious, while the iterative operators can be constructed by a repeated applications of the functions, Lemma 4.5 holds for them. For graph problems below, we use the following encoding of graphs into strings — Let graph $G = (V, E)$ with $|V| = n$. Assume that the vertices in V are numbered from 0 to $n - 1$. Define $gcode(G)$ to be the string of length n^2 such that bit $\langle i, j \rangle$, for $0 \leq i, j < n$, is 'on' iff edge $(i, j) \in E$. From now on, when we refer to a string as graph, we will mean the graph encoded by the string. Similarly, reference to a bit as edge will mean the edge encoded by the bit.

Example 1 : Hamiltonian Cycle Problem. Let

$$HAM \stackrel{\text{def}}{=} \{x \mid x \text{ is a directed graph containing a Hamiltonian cycle}\}$$

The relation witnessing HAM, R_{HAM} is : $xR_{HAM}s$ iff s is a subgraph of x and is a Hamiltonian cycle. R_{HAM} is admissible with $sol-len_{R_{HAM}}(x) = |x|$.

Theorem 4.14 R_{HAM} is universal.

Proof. To prove this, we show that R_{HAM} is S -universal, where S is the set of graphs that have the edge $\langle 0, 1 \rangle$ present in every solution of the graph.

Define $join_{R_{HAM}}(x, y) \stackrel{\text{def}}{=} z$ where z is obtained as follows : delete edge $\langle 0, 1 \rangle$ from both x and y , introduce edges from the vertex number 0 of x to the vertex number 1 of y and from the vertex number 0 of y to the vertex number 1 of x and finally renumber the vertices of x and y in the following way : let V be the number of vertices in x , then the vertex number 1 of x is assigned the number $V + 1$, the rest of the vertices of x retaining their number. For y , the vertex number 1 retains its number while the rest

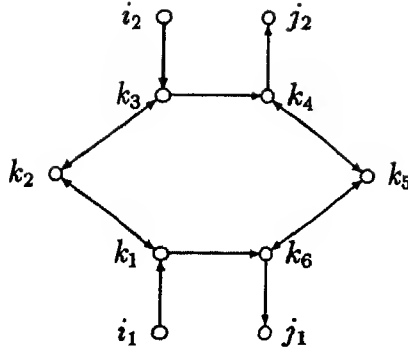


Figure 4.1: The modified part of the graph output by the negation operator for R_{HAM}

of them will be renumbered by adding V to their numbers. The two new edges will be present in every solution of z , thus, in particular, the edge $\langle 0, 1 \rangle$ will be present in all the solutions of z .

Define $neg_{HAM}(x, e_1, e_2) \stackrel{\text{def}}{=} z$ where z is obtained as follows : Let $e_1 = \langle i_1, j_1 \rangle$ and $e_2 = \langle i_2, j_2 \rangle$. z has six more vertices k_1, k_2, k_3, k_4, k_5 and k_6 with the extra edges $\langle i_1, k_1 \rangle, \langle k_1, k_2 \rangle, \langle k_2, k_3 \rangle, \langle k_3, k_4 \rangle, \langle k_4, k_5 \rangle, \langle k_5, k_6 \rangle, \langle k_6, j_1 \rangle, \langle i_2, k_3 \rangle, \langle k_3, k_2 \rangle, \langle k_2, k_1 \rangle, \langle k_1, k_6 \rangle, \langle k_6, k_5 \rangle, \langle k_5, k_4 \rangle$ and $\langle k_4, j_2 \rangle$. Edges e_1 and e_2 are deleted from z (see Figure 4.1). Edge $\langle i_1, k_1 \rangle$ corresponds to e_1 and $\langle i_2, k_3 \rangle$ to e_2 . It is easy to see that the new vertices can be covered only by either choosing edges $\langle i_1, k_1 \rangle, \langle k_1, k_2 \rangle, \langle k_2, k_3 \rangle, \langle k_3, k_4 \rangle, \langle k_4, k_5 \rangle, \langle k_5, k_6 \rangle, \langle k_6, j_1 \rangle$ or $\langle i_2, k_3 \rangle, \langle k_3, k_2 \rangle, \langle k_2, k_1 \rangle, \langle k_1, k_6 \rangle, \langle k_6, k_5 \rangle, \langle k_5, k_4 \rangle, \langle k_4, j_2 \rangle$. The rest of the solution remains the same. A suitable renumbering of the vertices can be easily worked out to ensure that the edge $\langle 0, 1 \rangle$ is present in every solution of z .

Define $block_{R_{HAM}}$ as in the Figure 4.2. It can be easily seen to satisfy the required properties.

□

Example 2 : Clique Problem. Let

$$\text{CLIQUE} \stackrel{\text{def}}{=} \{ \langle x, r \rangle \mid \text{undirected graph } x \text{ has an } r\text{-clique} \}$$

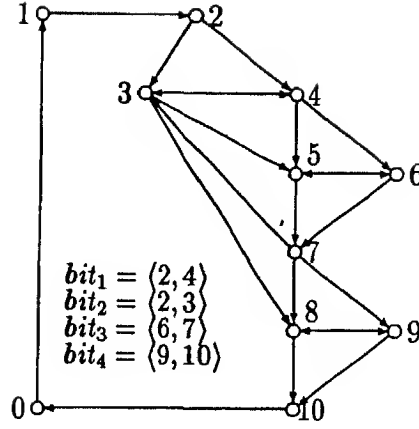


Figure 4.2: The instance $block_{R_{HAM}}$

The relation R_{CLIQUE} is : $\langle x, r \rangle R_{CLIQUE} s$ iff $|s| = n$, where n is the number of vertices in x and s has exactly r bits 'on' corresponding to the vertices in r -clique. R_{CLIQUE} is admissible with $sol-len_{R_{CLIQUE}}(\langle x, r \rangle) = n$.

Theorem 4.15 R_{CLIQUE} is universal.

Proof. We prove this by showing that R_{CLIQUE} is S -universal where S is the set of instances $\langle x, r \rangle$ that have at most an r -clique.

Define $join_{R_{CLIQUE}}(\langle x, r_1 \rangle, \langle y, r_2 \rangle) \stackrel{\text{def}}{=} \langle z, r_1 + r_2 \rangle$, graph z is obtained by renumbering all the vertices of y to make them different from those of x and then joining every vertex of y with every vertex of x . Since x and y can have at most r_1 -clique and r_2 -clique respectively, z will have at most $r_1 + r_2$ -clique.

Define $neg_{R_{CLIQUE}}(\langle x, r \rangle, i, j) = \langle z, r + 1 \rangle$. Graph z is obtained as follows — Add two new vertices k_1, k_2 and the following edges to x : vertex k_1 is joined to j as well as every vertex adjacent to j except i and vertex k_2 is joined to i as well as every vertex adjacent to i except j . Since x has at most an r -clique, any solution of z must have exactly one of the vertices k_1 or k_2 in the clique. Vertex k_1 will be present in the clique

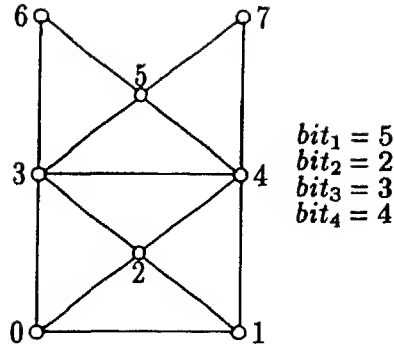


Figure 4.3: Graph for the instance $block_{R_{CLIQUE}}$

for exactly those solutions in which j is present while i is not and vertex k_2 will be present in the clique for exactly those solutions in which i is present while j is not.

Define instance $block_{R_{CLIQUE}} = \langle \text{graph in Figure 4.3}, 3 \rangle$. It can easily be seen to satisfy the required properties.

□

Example 3 : Partition Problem. This problem has r numbers as input and the solution is a division of the numbers in two sets having equal sum. We consider the following well known variation of the problem. Let

$PAR \stackrel{\text{def}}{=} \{ \langle n_1, n_2, \dots, n_r, m \rangle \mid \text{there is a subset of the set } \{n_1, \dots, n_r\} \text{ having sum } m \}$

Define relation R_{PAR} as : $\langle n_1, \dots, n_r, m \rangle R_{PAR} s$ iff $|s| = r$ and the set of numbers $\{n_i \mid s^i = 1\}$ has sum m .

Theorem 4.16 R_{PAR} is universal.

Proof. Define $join_{R_{PAR}}(x, y) \stackrel{\text{def}}{=} z$, where $x = \langle n_1, \dots, n_r, m_1 \rangle$, $y = \langle l_1, \dots, l_s, m_2 \rangle$ and $z = \langle n_1, \dots, n_r, l_1 * nsum, \dots, l_s * nsum, m_1 + m_2 * nsum \rangle$ where $nsum = 1 + \sum_{i=1}^r n_i$.

Define $neg_{R_{PAR}}(x, i, j) \stackrel{\text{def}}{=} z$, where $x = \langle n_1, \dots, n_r, m \rangle$ and $z = \langle l_1, \dots, l_r, m' \rangle$. For each k , $k \neq i$ or j , $l_k = n_k$, $l_i = n_i + nsum$, $l_j = n_j + nsum$ and $m' = m + nsum$, $nsum$ is defined as before.

To show that Lemma 4.5 holds for these functions is a bit tricky (one needs to choose the set S carefully and then show that R_{PAR} is S -universal by proving that the length constraints hold for the above two functions over the set S). However, the iterative versions of the two operators can be constructed by generalizing the above definitions also.

Define $iter-join_{R_{PAR}}(\langle x_1, \dots, x_k \rangle, k) \stackrel{\text{def}}{=} z$, where $x_i = \langle n_{i,1}, \dots, n_{i,r_i}, m_i \rangle$ for $1 \leq i \leq n$ and $z = \langle n_{1,1}, \dots, n_{1,r_1}, n_{2,1} * nsum, \dots, n_{2,r_2} * nsum, \dots, n_{k,1} * nsum^{k-1}, \dots, n_{k,r_k} * nsum^{k-1}, m_1 + \dots + m_k * nsum^{k-1} \rangle$ and $nsum = \max_{1 \leq i \leq k} \{1 + \sum_{j=1}^{r_i} n_{i,j}\}$. Similarly for the equivalence operator.

Define $block_{R_{PAR}} \stackrel{\text{def}}{=} \langle 8, 6, 3, 5, 1, 1, 1, 13 \rangle$ with $bit_1 = 3$, $bit_2 = 4$, $bit_3 = 5$, $bit_4 = 6$.

□

4.6.2 k -Creative sets

Apart from these natural sets, obvious relations of existing k -creative sets in NP can also be shown to be universal. Joseph and Young have defined the following class of k -creative sets in NP [10] —

$$K_f^k \stackrel{\text{def}}{=} \{f(i) \mid M_i \text{ accepts } f(i) \text{ within } |i| \cdot |f(i)|^k + |i| \text{ steps}\}$$

where f is polynomial time computable, one-one, honest and $k > 0$.

For these sets, the most obvious relations witnessing them will be the following : $x R_{k,f} i \# w$ iff $f(i) = x$ and w is an accepting computation of M_i on x .

Theorem 4.17 *Relation $R_{k,f}$ is universal.*

Proof. Let the function $iter\text{-}join_{R_{k,f}}(\langle x_1, \dots, x_n \rangle, n) \stackrel{\text{def}}{=} f(g_n(x_1, x_2, \dots, x_n))$ where TM $M_{g_n(x_1, \dots, x_n)}$, on input z , guesses the string $i_1\#w_1\#\#i_2\#w_2\#\#\dots\#\#i_n\#w_n$ and accepts iff for every k less than or equal to n , $f(i_k) = x_k$ and w_k is an accepting computation of M_{i_k} on x_k . One can easily ensure that the above guess string is written in some fixed bit positions in the accepting computation of $M_{g_n(x_1, \dots, x_n)}$. This enables one to compute $iter\text{-}join\text{-}seq_{R_{k,f}}$ in polynomial time.

Define $iter\text{-}equ_{R_{k,f}}(x, n, \langle i_1, \dots, i_n \rangle, \langle j_1, \dots, j_n \rangle) \stackrel{\text{def}}{=} f(h_{2n+1}(x, i_1, \dots, i_n, j_1, \dots, j_n))$ where TM $M_{h_{2n+1}(x, i_1, \dots, i_n, j_1, \dots, j_n)}$ on input z , guesses the string $i\#w$ and accepts iff $f(i) = x$, w is an accepting computation of M_i on x and for every k less than or equal to n , i_k^{th} and j_k^{th} bits of the string $i\#w$ are same. The $iter\text{-}equ\text{-}seq_{R_{k,f}}$ can be computed by ensuring that h_{2n+1} satisfies the same conditions as g_n above.

The instance $block_{R_{k,f}}$ is very trivial : $block_{R_{k,f}} \stackrel{\text{def}}{=} f(i_0)$, where M_{i_0} , on input z , guesses a string of length 4 and accepts only those guesses such that the solution set of $f(i_0)$ when restricted to these four bits satisfies the constraints as required in the definition 4.3. \square

4.7 Examples of non-universal relations

Relations witnessing the problems believed to be non-NP-complete as well as some relations witnessing NP-complete languages can be shown to be non-universal.

Example 1 : Horn Clause Problem. A horn-clause is a disjunction of boolean variables with at most one of them occurring positively. Define

$$\text{HORN} \stackrel{\text{def}}{=} \{x \mid x \text{ is a satisfiable horn-formula}\}$$

The relation R_{HORN} is : xR_{HORN} iff x is a horn-formula and xR_{SAT} .

Theorem 4.18 R_{HORN} is not universal.

Proof. Let set $W = \{10, 01\}$. Suppose x_0 is a horn-formula with α a sequence such that $Mask(sol_{R_{HORN}}(x_0), \alpha) = W$. Let $\alpha = i_0, j_0$.

Define function $reduce \stackrel{\text{def}}{=} \lambda x, i. z$, where $sol_{R_{HORN}}(z) = Mask(sol_{R_{HORN}}(x), \beta)$ for $\beta = 1, \dots, i-1, i+1, \dots, n$ with n being the number of variables in x . Formula z is obtained by taking the disjunction of $x_{v_i=T}$ and $x_{v_i=F}$, where $x_{v_i=T}$ and $x_{v_i=F}$ denote the formulae obtained from x by setting variable v_i to true and false respectively. It is easy to see that if x is a horn-formula then z will also be a horn-formula.

Using $reduce$ function iteratively, we get rid of all the variables in x_0 except variable numbers i_0 and j_0 . Let y be the resultant instance. So, $sol_{R_{HORN}}(y) = W$ and y will still be a horn-formula. It is easy to see that using two variables there is no way of constructing a horn-formula having the above solution set which is a contradiction. Therefore, by Corollary 4.9, R_{HORN} is not universal. \square

Example 2 : Graph Isomorphism. Define

$$GISO \stackrel{\text{def}}{=} \{\langle x, y \rangle \mid \text{graphs } x \text{ and } y \text{ are isomorphic}\}$$

The relation R_{GISO} is : $\langle x, y \rangle R_{GISO} s$ iff s encodes an isomorphism of x and y in the following way — $s^i = 1$ with $i = \langle j, k \rangle$ iff vertex j of x is mapped to vertex k of y . Clearly $sol-len_{GISO} = \lambda \langle x, y \rangle. \langle n-1, n-1 \rangle$ where n is the number of vertices in x .

Theorem 4.19 R_{GISO} is not universal.

Proof. Let $W = \{11, 10, 01\}$ and assume that there is an instance $z_0 = \langle x_0, y_0 \rangle$ and a sequence α such that $Mask(sol_{R_{GISO}}(z_0), \alpha) = W$. Let $\alpha = \langle i_1, j_1 \rangle, \langle i_2, j_2 \rangle$.

For any instance $z = \langle x, y \rangle$, let $F(z)$ be the set of all isomorphic mappings between graphs x and y . Consider all the mappings in $F(z)$ that map vertex i of x to vertex j of y . Call this set $F(z, i, j)$. So, $F(z, i, j) = \{f \mid f \in F(z) \wedge f(i) = j\}$. Under mappings

in $F(z, i, j)$, any vertex k of x will be mapped to a set of vertices of y . Let this set be called $V(z, i, j, k)$. So, $V(z, i, j, k) = \{l \mid f \in F(z, i, j) \text{ \& } f(k) = l\}$.

Claim : For any $z = \langle x, y \rangle$, i, j_1, j_2 and k : if the sets $F(z, i, j_1)$ and $F(z, i, j_2)$ are non-empty then the sets $V(z, i, j_1, k)$ and $V(z, i, j_2, k)$ have the same number of elements, i.e., $|V(z, i, j_1, k)| = |V(z, i, j_2, k)|$.

Proof. Let $f_1 \in F(z, i, j_1)$ and $f_2 \in F(z, i, j_2)$. Define $g = f_2 \circ f_1^{-1}$. g is an automorphism of graph y with $g(j_1) = j_2$. Similarly, $g^{-1} = f_1 \circ f_2^{-1}$ is also an automorphism of graph y with $g^{-1}(j_2) = j_1$. These two mappings satisfy the following properties —

1. Both g and g^{-1} are one-one and onto.
2. For any $h \in F(z)$, both $g \circ h$ and $g^{-1} \circ h$ are in $F(z)$.
3. For any two $h_1, h_2 \in F(z)$, if $g \circ h_1 = g \circ h_2$ then $h_1 = h_2$. This will hold for g^{-1} as well (since both of them are one-one).
4. For $h \in F(z, i, j_1)$, $g \circ h \in F(z, i, j_2)$. Similarly, for $h \in F(z, i, j_2)$, $g^{-1} \circ h \in F(z, i, j_1)$.

Any mapping f in $F(z, i, j_2)$ can be written as $g \circ f'$ for some $f' \in F(z, i, j_1)$ ($f' = g^{-1} \circ f$). Similarly, any mapping f in $F(z, i, j_1)$ can be written as $g^{-1} \circ f'$ for some $f' \in F(z, i, j_2)$ ($f' = g \circ f$). So, $g(F(z, i, j_1)) = F(z, i, j_2)$ and $g^{-1}(F(z, i, j_2)) = F(z, i, j_1)$. Therefore,

$$\begin{aligned}
 g(V(z, i, j_1, k)) &= \{g(l) \mid l \in V(z, i, j_1, k)\} \\
 &= \{g(l) \mid f \in F(z, i, j_1) \text{ \& } f(k) = l\} \\
 &= \{(g \circ f)(k) \mid f \in F(z, i, j_1)\} \\
 &= \{(g \circ f)(k) \mid g \circ f \in F(z, i, j_2)\} \\
 &= \{f(k) \mid f \in F(z, i, j_2)\}
 \end{aligned}$$

$$= V(z, i, j_2, k)$$

Since g is one-one, $|V(z, i, j_1, k)| = |V(z, i, j_2, k)|$. \square

We have $Mask(sol_{R_{GISO}}(z_0), \alpha) = \{11, 01, 10\}$ with $\alpha = \langle i_1, j_1 \rangle, \langle i_2, j_2 \rangle$. So, there will be an isomorphic mapping between x_0 and y_0 in which vertex i_1 of x_0 is *not* mapped to vertex j_1 of y_0 . Suppose it is mapped to vertex j_3 of y_0 . So, in any mapping of $F(z_0, i_1, j_3)$, vertex i_2 of x_0 *must* be mapped to vertex j_2 of y_0 . Thus, $V(z, i_1, j_3, i_2) = \{j_2\}$. Consider the case when vertex i_1 is mapped to vertex j_1 . There will exist two different mappings in $F(z, i_1, j_1)$ such that in one of them vertex i_2 is mapped to vertex j_2 and in the other vertex i_2 is not mapped to vertex j_2 . Thus $|V(z, i_1, j_1, i_2)| \geq 2$. This contradicts the above claim. Therefore, by Corollary 4.9, R_{GISO} is not universal. \square

Relations witnessing some NP-complete languages can also be shown to be non-universal. However, there will be a difference between the reasons for non-universality of such relations and that of the above two relations. We first give an example of a non-universal relation witnessing an NP-complete language and then will try to capture this difference by generalizing the definition of universal relations.

Example 3 : Simple Max Cut Problem. Let

$$SMC \stackrel{\text{def}}{=} \{\langle x, r \rangle \mid x = (V, E) \text{ is an undirected graph having at least an } r\text{-cut}\}$$

Define relation R_{SMC} as : $\langle x, r \rangle R_{SMC} s$ iff s is a subset of the vertices of x giving at least an r -cut.

Theorem 4.20 R_{SMC} is not universal.

Proof. There will be no instance such that its solution set, restricted to some sequence, is $\{1\}$. This follows because if s is a solution of $\langle x, r \rangle$ then so will be \bar{s} . Therefore, by Corollary 4.9, R_{SMC} is not universal. \square

One can construct non-universal relations for *every* language in NP : Take any relation for the language and modify it so that if s is a solution for some instance in the original relation then both s and \bar{s} are the solutions for the instance in the new one. That it is non-universal follows from argument identical to the above Theorem. However, the reason for the non-universality of all such relations is that they have redundant solutions : if s is a solution then \bar{s} essentially represents same solution in a different form. This is definitely *not* the case with the two non-universal relations in examples 1 and 2. Therefore, one would like to distinguish between these two types of non-universalities. Towards this, we define the concept of *refinement* of a relation and using this, a generalized universal relation.

In a refinement of a relation we would want to *compress* equivalent solutions of an instance into a single solution. We also would want to ensure that non-equivalent solutions remain different. In the following definition, we define a polynomial time function which maps many solutions of an instance into one and use it to achieve the required compression.

Definition 4.21 Relation R' is *refinement* of the relation R if there is a polynomial time computable binary function h such that the following conditions hold for every instance x —

Let $W_s = \{t \mid h(x, t) = s\}$.

1. $(\forall s)[[W_s \neq \emptyset] \Rightarrow [s \in W_s]]$.
2. $(\forall s)(\forall y)[[W_s \subseteq \text{sol}_R(y)] \vee [W_s \cap \text{sol}_R(y) = \emptyset]]$.

And $xR's$ iff xRs and $h(x, s) = s$.

Function h is a many-one function and R' admits only those solutions in $\text{sol}_R(x)$ that are mapped to themselves by h . Condition 1 ensures that no solutions are 'missed

out' while the condition 2 tries to ensure that only equivalent solutions are compressed. It says that if a set of strings is compressed into one then for every instance either all the strings in the set are solutions of the instance or none is.

Definition 4.22 Let R be an admissible relation. R is a *generalized universal* relation if there is a refinement R' of R such that R' is universal.

Every universal relation will be generalized universal also, as relation R is a refinement of itself. The following two propositions immediately follow.

Proposition 4.23 Set A has a generalized universal relation iff it has a universal relation.

Proposition 4.24 Let R be a generalized universal relation witnessing the set A . Then A is NP-complete.

One can quite easily define a refinement of R_{SMC} which will be universal.

Theorem 4.25 R_{SMC} is generalized universal.

Proof. Define the following refinement of R_{SMC} : $\langle x, r \rangle R'_{SMC} s$ iff s is a subset of vertices of x giving at least an r -cut and s does not contain vertex number 0, i.e., $h(\langle x, r \rangle, 1s) = h(\langle x, r \rangle, 0s) = 0s$ for every s .

We define the operators on the set of instances $\langle x, r \rangle$ that have *at most* an r -cut.

Define $join_{R'_{SMC}}(\langle x, r_1 \rangle, \langle y, r_2 \rangle) \stackrel{\text{def}}{=} \langle z, r_1 + r_2 \rangle$, where z is the graph obtained by collapsing the vertex number 0 of x and y into a single vertex also numbered 0 in z . The rest of the vertices remain as they are, except for a renumbering.

Define $equ_{R'_{SMC}}(\langle x, r \rangle, \langle i, j \rangle) \stackrel{\text{def}}{=} \langle z, r + 2 \rangle$, where graph z is obtained by adding a new vertex k to x and joining both the vertices i and j to it. This ensures that either both i and j must be chosen or none.

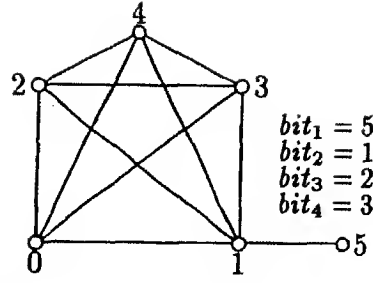


Figure 4.4: Graph for the instance $block_{R'_{SMC}}$

Define $block_{R'_{SMC}} \stackrel{\text{def}}{=} \langle \text{graph in Figure 4.4, 7} \rangle$.

□

Proposition 4.26 *Relation R_{HORN} and R_{GISO} are not generalized universal.*

Proof Sketch. Relations R_{HORN} and R_{GISO} do *not* have any refinement except themselves as for every solution s for some instance x there will be an instance y whose solution set contains only s . Therefore, neither is a generalized universal relation.

□

4.8 Structural properties of the two operators

In this section, we investigate the structural properties, e.g., paddability, d-self-reducibility, of the operators *join* and *equivalence*.

4.8.1 The Join operator

The join operator is closely related to paddability. For join operator of a relation R witnessing the set A , we have, $join_R(x, y) \in A$ iff $x \in A \wedge y \in A$. Thus, one can pad any instance using this operator. The following theorem immediately follows.

Theorem 4.27 *Let R be a relation witnessing the set A . If R has a join operator then both A and \bar{A} contain an infinite PTIME subset, i.e., they are not p -immune.*

Proof. Let $x_0 \in A$ and $y_0 \notin A$ (the case when $A = \emptyset$ or Σ^* is trivial). We first define two functions :

$$m\text{-join}_R(x, n) \stackrel{\text{def}}{=} \begin{cases} \text{join}_R(m\text{-join}_R(x, n-1), x) & \text{if } n > 1 \\ x & \text{if } n = 1 \end{cases}$$

and, $si\text{-join}_R(x) \stackrel{\text{def}}{=} m\text{-join}_R(x, n)$, where $n = \mu_m \{|m\text{-join}_R(x, m)| > |x|\}$

The length of a solution of instance $m\text{-join}_R(x, n)$ will be at least n and since the length of a solution of an instance is bounded by a polynomial in the length of the instance, $|m\text{-join}_R(x, n)| \geq p^{-1}(n)$ for some polynomial p . Therefore, function $si\text{-join}_R$ will be computable in polynomial time as at most a polynomial number of applications of $m\text{-join}_R$ are required to compute it. Now define the sets $A_1 = \{si\text{-join}_R^i(x_0) \mid i \geq 0\}$ and $A_2 = \{si\text{-join}_R^i(y_0) \mid i \geq 0\}$, where $si\text{-join}_R^i(x) = si\text{-join}_R(si\text{-join}_R^{i-1}(x))$ if $i > 0$; x if $i = 0$. Both A_1 and A_2 will be infinite and polynomial time recognizable with $A_1 \subseteq A$ and $A_2 \subseteq \bar{A}$. \square

The following theorem relates paddability to the join operator.

Theorem 4.28 *Let R be a relation witnessing the set A . If R has an iterative join operator with the function $iter\text{-join}_R$ polynomially invertible, then A is paddable.*

Proof. Let x_0 and x_1 be two different instances of A with $x_0, x_1 \in A$. Define function $pad(x, y) \stackrel{\text{def}}{=} iter\text{-join}_R(\langle x, x_{y^1}, x_{y^2}, \dots, x_{y^{|y|}} \rangle, |y| + 1)$. Function pad is computable in polynomial time and is invertible as well. Moreover, $x \in A$ iff $pad(x, y) \in A$ for any y . Therefore, pad is a padding function for A . \square

4.8.2 The Equivalence operator

The equivalence operator restricts the solution space of any instance. So, we can use it to restrict the solution space in such a way that the resulting instance can only have a particular solution, if at all. This property is similar to d -self-reducibility. The following theorem shows how this property can be used to obtain d -self-reducibility for the set.

Theorem 4.29 *Let R be a relation witnessing the set A . If R has an iterative equivalence operator with the function $iter-equ_R$ invertible in polynomial time, then A is d -self-reducible.*

Proof. The intuitive idea behind how a d -self-reducing tree can be obtained for an instance x is as follows. For each k , $2 \leq k \leq sol-len_R(x)$, there is a subtree with bit positions 1 and k treated as *chosen*. The nodes of every such subtree are obtained by successively equating solution bits other than 1 and k to one of these two bits. At the leaf we will have only those nodes whose solutions depend only on the assignments to bits 1 and k , and therefore, the test whether such nodes are in the set A can be made in polynomial time. Now, we give the construction in detail.

For any y , say that y is *properly invertible* if

1. $iter-equ_R^{-1}(y)$ is defined, and
2. let $y = iter-equ_R(x, m, \langle i_1, \dots, i_m \rangle, \langle j_1, \dots, j_m \rangle)$, then the following conditions hold —
 - (i) $m \leq sol-len_R(x) - 2$.
 - (ii) $2 \leq j_1 < j_2 < \dots < j_m \leq m + 2$.
 - (iii) $1 \leq |\{i_1, i_2, \dots, i_m\}| \leq 2$.

$$(iv) \{i_1, i_2, \dots, i_m\} \cap \{j_1, j_2, \dots, j_m\} = \emptyset.$$

$$(v) i_k < j_k \text{ for every } k, 1 \leq k \leq m.$$

Define the relation $<$ as : $y < z$ iff y is properly invertible and one of the following two cases hold :

Case 1 : z is properly invertible with

1. $z = \text{iter-equ}_R(x, m, \langle i_1, \dots, i_m \rangle, \langle j_1, \dots, j_m \rangle)$
2. $m < \text{sol-len}_R(x) - 2$
3. $y = \text{iter-equ}_R(x, m+1, \langle i_1, \dots, i_{m+1} \rangle, \langle j_1, \dots, j_{m+1} \rangle)$.

Case 2 : z is not properly invertible and $y = \text{iter-equ}_R(z, 1, i_1, j_1)$.

Define the partial ordering $<^*$ as the reflexive and transitive closure of $<$. This partial ordering is OK since the length of any $<^*$ -decreasing chain starting from z is less than $\text{sol-len}_R(z) + \text{sol-len}_R(x)$, where $x = \pi_1(\text{iter-equ}_R^{-1}(z))$, which is bounded by a polynomial in $|z|$. The size of every element in this chain is also bounded by a polynomial in $|z|$.

Define the d-self-reducing TM M recognizing A as —

On input z ,

1. if z is not properly invertible then compute the set

$$Q = \{\text{iter-equ}_R(z, 1, 1, 2), \text{iter-equ}_R(z, 1, 1, 3), \text{iter-equ}_R(z, 1, 2, 3)\}$$

and accept iff $Q \cap A \neq \emptyset$.

2. if z is properly invertible with

$$(a) z = \text{iter-equ}_R(x, m, \langle i_1, \dots, i_m \rangle, \langle j_1, \dots, j_m \rangle)$$

(b) $m < \text{sol-len}_R(x) - 2$

then let $I = \{1\} \cup \{i_1, \dots, i_m\}$ (by definition, $1 \leq |I| \leq 2$). If $|I| = 1$ then choose the *smallest* two numbers i and j , $i < j$, not in the set $I \cup \{j_1, \dots, j_m\}$ and compute the set

$$\begin{aligned} Q = & \{ \text{iter-equ}_R(x, m+1, \langle i_1, \dots, i_m, 1 \rangle, \langle j_1, \dots, j_m, i \rangle), \\ & \text{iter-equ}_R(x, m+1, \langle i_1, \dots, i_m, 1 \rangle, \langle j_1, \dots, j_m, j \rangle), \\ & \text{iter-equ}_R(x, m+1, \langle i_1, \dots, i_m, i \rangle, \langle j_1, \dots, j_m, j \rangle) \} \end{aligned}$$

If $|I| = 2$ then let $I = \{1, i\}$ (say) and choose the *smallest* number j not in the set $I \cup \{j_1, \dots, j_m\}$ and compute the set

$$\begin{aligned} Q = & \{ \text{iter-equ}_R(x, m+1, \langle i_1, \dots, i_m, 1 \rangle, \langle j_1, \dots, j_m, j \rangle), \\ & \text{iter-equ}_R(x, m+1, \langle i_1, \dots, i_m, i \rangle, \langle j_1, \dots, j_m, j \rangle) \} \end{aligned}$$

Accept iff $Q \cap A \neq \emptyset$.

3. if z is properly invertible with

(a) $z = \text{iter-equ}_R(x, m, \langle i_1, \dots, i_m \rangle, \langle j_1, \dots, j_m \rangle)$

(b) $m = \text{sol-len}_R(x) - 2$

then let $\{1, i\} = \{1, 2, \dots, m+2\} - \{j_1, \dots, j_m\}$, $S = \{1\} \cup \{j \mid (\exists k) i_k = 1 \wedge j_k = j\}$ and $T = \{i\} \cup \{j \mid (\exists k) i_k = i \wedge j_k = j\}$. Accept iff there is a solution s of x such that $s^k = s^l$ for every $k, l \in S$ or $k, l \in T$. This can be tested in polynomial time as one has to check for only four possible solutions of x .

That the above TM accepts the language A is easy to verify : using induction one can show that at each level m of the self-reducing tree the following holds. Let

y_1, y_2, \dots, y_k be the nodes at the level m . Then for each node y_i there exists a polynomial time computable sequence α_i such that

$$\bigcup_{i=1}^k \{Mask(sol_R(y_i, \alpha_i))\} = sol_R(x)$$

and first $m + 2$ bits of each of the sets $Mask(sol_R(y_i, \alpha_i))$ have at most four different assignments. Thus, A is d-self-reducible. \square

4.9 The universal and generalized universal relations

We have seen that if a set has a universal relation then it is NP-complete. Can we say that *every* NP-complete set has a universal relation (or equivalently, a generalized universal relation) ? An affirmative answer will obviously imply $P \neq NP$ as finite sets can not have a universal relation (this follows from Theorem 4.27) and so it is not an easy question to answer. However, all sets in the p-isomorphism degree of SAT can be easily shown to have a universal relation.

Theorem 4.30 *Let A be p-isomorphic to SAT. Then there is a universal relation R witnessing A .*

Proof Sketch. Let f be the polynomial time isomorphism such that $x \in A$ iff $f(x) \in SAT$. Define relation R as : xRw iff $f(x)R_{SAT}w$ and function

$$iter-join_R(\langle x_1, \dots, x_n \rangle, n) \stackrel{\text{def}}{=} f^{-1}(iter-join_{R_{SAT}}(\langle f(x_1), \dots, f(x_n) \rangle, n))$$

The other functions can be defined similarly. \square

All the sets having a universal relation can be shown to be complete for NP under honest reductions (the following theorem). Therefore, if the NP-complete degree has

more than one honest degrees then there will be NP-complete sets having no universal relation.

Theorem 4.31 *Let R be a universal relation witnessing the set A . Then A is complete for NP under honest polynomial time reductions.*

Proof. Let f be the polynomial time reduction of SAT to A . Define function $g(x) = \text{si-join}_R(f(x))$ where si-join_R is the function defined in the proof of Theorem 4.27. Function g is a size-increasing reduction of SAT to A . \square

We know that there are sets having no universal relation (e.g., finite sets and all sets in P under the assumption $P \neq NP$) and that there are sets having a universal as well as a non-universal relation. Are there sets such that all admissible relations witnessing them are universal. Obviously not, as every set has a non-universal relation. But, as we have seen, some of these non-universal relations are generalized universal. So, we ask, if there is an NP-complete set such that *every* admissible relation witnessing the set is generalized universal? In particular, is every admissible relation witnessing SAT is generalized universal? We do not know an answer to this. We can only show that if it is true for SAT then it is true for *every* set p-isomorphic to SAT.

Theorem 4.32 *If every admissible relation witnessing SAT is generalized universal then every admissible relation witnessing every set p-isomorphic to SAT, is generalized universal.*

Proof Sketch. Let R be some admissible relation witnessing some set A which is p-isomorphic to SAT and f be the isomorphism between the two. By moving back and forth using f , as in the proof of Theorem 4.30, one can construct the operators for some refinement of R . \square

Chapter summary

The concept of universal relations suggests the following characterization of the sets in NP-complete degree : every NP-complete set is witnessed by a universal relation. As we have seen, if this is indeed a characterization then $P \neq NP$. All paddable NP-complete sets as well as classes of k -creative sets are witnessed by a universal relation thus providing some evidence that the characterization holds. A related question here is that whether every relation witnessing any NP-complete set is generalized universal. In case this has a positive answer then, besides showing $P \neq NP$, it provides a way of proving sets to be non-NP-complete as we have shown that there are relations that are not generalized universal.

Chapter 5

Concluding Remarks

In the preceding chapters, we have defined two sufficient properties for a set to be NP-complete. We have also seen that all existing NP-complete sets seem to satisfy each of these properties and that no finite set can satisfy either. Thus, either of the properties may be considered as a possible alternative characterization of NP-complete sets and if so then $P \neq NP$.

The first property is called (p, CM_{NP}) -creativity and is a generalization of the earlier definition of creativity for NP, namely, k -creativity. We have seen that every lpr-paddable NP-complete set will be (p, CM_{NP}) -creative and natural complete sets seem to be indeed lpr-paddable. We have also defined a subclass of NP-complete sets, namely, the class $\{Smp(A) \mid A \text{ is NP-complete}\}$, no member of which is lpr-paddable. However, even the sets in this class turn out to be (p, CM_{NP}) -creative.

We were led to the notion of \leq_m^{pr} -reducibility through certain technical considerations. However, this idea may be of independent interest as in some way it formalizes many natural reductions which work on successive 'local' information.

We have also obtained new definitions of creativity for classes PSPACE, EXP, NEXP and r.e. For the definitions for EXP, NEXP and r.e., we show that they are

equivalent to the existing definitions in the literature. For the classes EXP and NEXP, it turns out that creative sets characterize many-one complete sets while for PSPACE they characterize logspace-complete sets. For logspace-complete sets in PSPACE, we also show that they are complete under one-one and size-increasing reductions.

Using creativeness, we have shown for classes varying from DLOG to NEXP that 1-L-complete sets for these classes are p-isomorphic. The p-isomorphism of complete sets of various classes under more general reductions is a major open problem. The biggest obstacle in proving such results is a proof of polynomial time invertibility of such reductions. This problem is not there in the case of 1-L reductions as they can be easily inverted in polynomial time. For more general reducibilities, very little is known about the structure of the complete degree for various classes; it is not clear if the notion of creativeness will be helpful in exploring the isomorphism question of such degrees.

In chapter 4, we have defined a property, called the universal property, for relations witnessing sets in NP and have seen that if a set is witnessed by a universal relation then it is NP-complete. We have shown that every paddable NP-complete set as well as well known classes of k -creative sets are witnessed by universal relations. A universal relation has two operators associated with it, called join and equivalence. Standard relations of many natural NP-complete sets are universal, further, the associated operators in these cases turn out to be polynomial time invertible. We have shown that if the join operator is polynomial time invertible then the corresponding witnessed set will be paddable, and if the equivalence operator is polynomial time invertible then the set will be d-self-reducible. Thus, the existence of such invertible operators for natural complete sets may be considered as the explanation for their being paddable and d-self-reducible. For some sets in NP, believed to be non-complete, we show that their standard relations are not universal. This can be interpreted as showing that if

these sets turn out to be NP-complete then the reductions of languages in NP to these sets must be very different from the ones that exist for natural NP-complete sets.

We have arrived at the definitions of the two operators via combinatorial considerations. An interesting problem here is to reduce these two operators in terms of more atomic combinatorial properties. One possible approach is to formalize the notion of ‘local’ and ‘global’ constraints so that the solution set of any instance can be specified in terms such constraints. For example, the property that a graph is a *Hamiltonian cycle* can be captured by the following constraints : (1) both the indegree and outdegree of each vertex of the graph is exactly one (this is a ‘local’ property as it holds at every vertex) (2) the graph is connected (this is a ‘global’ property). Similarly, the property that a graph is a *k-clique* can be captured by the following constraints : (1) the degree of each vertex of the graph is either zero or k (a ‘local’ property) (2) the graph contains exactly $k(k-1)/2$ edges or alternately that the graph has exactly $n - k + 1$ components (both ‘global’ properties). Once a reasonably complete formalization of the idea of capturing solution sets in terms of a set of constraints is available, it will be interesting to see if the existence of the operators that we have defined can be explained in terms of certain characteristics of such sets of constraints.

Finally, one may try to relate the two sufficient properties that we have obtained in this thesis. It is easy to see that if the function $join_R$ is a pr-function then a set witnessed by universal relation R will be (p, CM_{NP}) -creative. However, we do not know whether the two properties are equivalent or whether one subsumes the other. It is difficult to perceive a meeting ground for them as they arise out of two entirely different kinds of considerations.

Bibliography

- [1] E. W. Allender. Isomorphisms and 1-L reductions. In *Proceedings of the Structure in Complexity Theory Conference*, pages 12–22. Springer Lecture Notes in Computer Science 223, 1986.
- [2] L. Berman. *Polynomial Reducibilities and Complete Sets*. PhD thesis, Cornell University, 1977.
- [3] L. Berman and J. Hartmanis. On isomorphism and density of NP and other complete sets. *SIAM Journal on Computing*, 1:305–322, 1977.
- [4] S. Cook. The complexity of theorem proving procedures. In *Proceedings of the Third ACM Symposium on Theory of Computing*, pages 151–158, 1971.
- [5] K. Ganesan and S. Homer. Complete problems and strong polynomial reducibilities. In *Proceedings of the Sixth Symposium on Theoretical Aspects of Computer Science*, pages 240–250. Springer Lecture Notes in Computer Science 349, 1988.
- [6] M. Garey and D. Johnson. *Computers and Intractability : A Guide to the Theory of NP-completeness*. Freeman, San Francisco, 1978.
- [7] S. Homer. On simple and creative sets in NP. *Theoretical Computer Science*, 47:169–180, 1986.

- [8] J. Díaz J. Balcázar and J. Gabarró. *Structural Complexity I*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1988.
- [9] N. Immerman J. Hartmanis and S. Mahaney. One-way log-tape reductions. In *Proceedings of the Nineteenth IEEE Symposium on Foundations of Computer Science*, pages 65–72, 1978.
- [10] D. Joseph and P. Young. Some remarks on witness functions for nonpolynomial and noncomplete sets in NP. *Theoretical Computer Science*, 39:225–237, 1985.
- [11] K. Ko and D. Moore. Completeness, approximation and density. *SIAM Journal on Computing*, 10:787–796, 1981.
- [12] D. Kozen. Indexing of subrecursive classes. *Theoretical Computer Science*, 11:277–301, 1980.
- [13] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York, 1967.
- [14] J. Wang. P-creative sets vs. p-completely creative sets. In *Proceedings of the Structure in Complexity Theory Conference*, pages 24–33, 1989.

Index

Following is the index of the notations and key definitions appearing in the thesis. The associated numbers denote the page at which they are defined.

admissible relation,	57	CONST,	12
α ,	57	$d_{iso}(A)$,	10
α, β ,	57	$d_m(A)$,	10
α^j ,	57	$d_{pr}(A)$,	20
$ \alpha $,	57	equivalence operator,	58
$\alpha + m$,	57	equ_R ,	58
$\alpha \mid \beta$,	57	$equ-seq_R$,	58
$block_R$,	59	exponentially honest,	35
$block_R^k$,	63	(F, I) -creative,	11
building block,	59	(F, I) -productive,	11
C_α ,	45	(F, I) -productive function,	11
CM_E ,	48	generalized universal relation,	79
CM_{NE} ,	48	iterative equivalence operator,	60
CM_{NP} ,	12	iterative join operator,	60
$CM_{NP(r)}$,	12	iterative negation operator,	66
CM'_{NP} ,	38	$iter-equ_R$,	60
CM_{PSPACE} ,	50		
CM_{RE} ,	48		

- $iter-equ-seq_R$, 60
- $iter-join_R$, 60
- $iter-join-seq_R$, 60
- $iter-neg_R$, 66
- $iter-neg-seq_R$, 66
- join operator, 58
- $join_R$, 58
- $join-seq_R$, 58
- K_{NP} , 15
- K'_{NP} , 38
- K_{PSPACE} , 50
- $\mathcal{L}(I)$, 10
- (lg, CM_{PSPACE}) -creative, 50
- LP_A , 22
- lpr-paddable, 22
- lpr-padding function, 22
- $Mask(S, \alpha)$, 57
- M_i , 10
- $m-join_R$, 81
- $M'_{j'}$, 37
- negation operator, 66
- neg_R , 66
- $neg-seq_R$, 66
- NPM , 11
- NPM_α , 44
- NPM^k , 11
- $Pad(n, i)$, 10
- (p, CM_E) -creative, 48
- (p, CM_{NE}) -creative, 48
- (p, CM_{NP}) -creative, 13
- $(p, CM_{NP(r)})$ -creative, 15
- (p, CM'_{NP}) -creative, 38
- ϕ_i , 10
- ϕ_i^t , 10
- p_i , 10
- (p, I) -creative, 11
- \leq_m^p , 10
- $\leq_{m,i}^p$, 10
- $\leq_{m,1}^p$, 10
- $\leq_{m,si}^p$, 10
- PM , 11
- (p, NPM) -creative, 11
- (p, NPM^k) -creative, 12
- (p, PM) -creative, 11
- (pr, CM_{NP}) -creative, 20
- PRF , 18
- pr-function, 18
- pr-immune, 27
- \leq_m^{pr} , 20
- pr-reduction, 20

- pr-simple, 27
- (rec, CM_{RE}) -creative, 48
- (rec, N) -creative, 11
- refinement (of a relation), 78
- RPA , 26
- $si\text{-}join_R$, 81
- S_{M_i} , 10
- $Smp(A)$, 30
- $sol\text{-}len_R$, 57
- $sol_R(x)$, 57
- solution, 57
- solution set, 57
- S -universal relation, 67
- t_α , 45
- T_{M_i} , 10
- universal relation, 62
- W_i , 10
- x^i , 56

CSE-1881-D-AGR-TOLV